Analysis and Implementation of a Method to Smooth Data Functions

An Adaptation of the Cubic Spline Method

Sarah Rohart

Master's thesis
2020:E85

LUND UNIVERSITY

Centre for Mathematical Sciences
Mathematics

Abstract

The goal of the thesis is to create a method that can be used to smooth data functions so that they are more easily readable and interpretable, through curve fitting. So as to do that, we adapt the smoothing spline method, which uses discrete data, to continuous data. In the smoothing spline method, we solve the variational problem of finding the minimizer of the following functional:

$$p\int_0^X y''(x)^2dx + (1-p)\sum_{i=1}^{m} w_i(y(x_i) - y_i)^2 \tag{1}$$

for all y in a suitable function space, with p the smoothing parameter controlling the relative importance of the smoothness of the curve in relation to the importance of the closeness of the curve to the data points. Those data points are the y_i adjoined with w_i weights letting us adjust the importance of a point according to how reliable the data is. It is known [1], if $p = 0$, that y is a least squares straight line, and if $p = 1$, that the minimizing function y is a third degree polynomial between the spline knots x_i, and the coefficients of the different polynomials have to match so that the curve is C^2 on the whole interval $[0, X]$ and satisfies a jump condition in the third derivative. These conditions for the minimizer give rise to a linear system involving an invertible matrix, and the problem is therefore solvable.

In this project we are not given discrete points as data points, but instead we are given a continuous function f which is defined on subintervals of the interval $[0, X]$. It is then natural to replace the least square sum in the variational problem for the smoothing spline by a least square integral, and so we minimize

$$p\int_0^X y''(x)^2dx + (1-p)\int_0^X w(x)(y(x) - f(x))^2dx \tag{2}$$

among all y in a suitable function space. Here, the weight function $w(x)$ is set to 0 in the intervals where f is not defined, and then we can extend f to a continuous function on the whole interval. We will focus on $w(x)$ being piece-wise constant, typically taking values 0 or 1. The parameter p controls the importance of the data in comparison to the smoothness of the function. The minimizer of the variational problem is expected to satisfy a fourth order linear inhomogeneous ODE, and in the project this ODE will be derived from the variational problem, and then it will be solved using ODE theory and numerical methods, using MATLAB.

Acknowledgments

First of all, I especially want to thank Sara Maad Sasane who has been my supervisor and guided me. She was a great help, reading and suggesting ways to improve it time after time, answering emails spontaneously and not losing hope despite personal issues on my side which delayed this thesis' publication, and thus this paper could not have been done without her.

Secondly, I want to thank Claus Führer, who has been an inspiration to do this thesis in this area - most of my Numerical Analysis background comes from lectures he's given. He has also helped with the numerical implementation, explaining some MATLAB algorithms.

Thirdly, I want to thank all my family, without whom my studying here in Lund wouldn't have been possible. Their support has been the basis without which this paper would have never seen the light of day.

Lastly, a special thanks to Mélanie Fournier, who has always been there for me during this process, and deserves a mention for all the support she has given me time after time.

Contents

Chapter 1

Introduction

One of the most commonly found mathematical problems in engineering and science is how to create a curve using known data points. One approach at solving those problems is the spline, a word that has its roots in East Anglian, referencing the devices used by wood-workers to create a smooth curve. The idea was that, in shipbuilding, thanks to the elasticity of the wood, one could apply weights - then called *ducks* - in certain points, called knots or control points, and the piece of wood would bend in such a way so that it used the least energy to bend between the knots, hence being the ”smoothest” possible.
In the same manner, and with almost exactly the same idea, splines today are functions, defined piece-wise by polynomials, such that, in the case of what is called interpolating curves, they go through all the knots. This idea was first introduced by Isaac Jacob Schoenberg, during the Second World War, which saved the need to measure thousands of points on planes, replacing it by a more efficient mathematical model. This model was only mathematical, and Bézier and de Casteljau, respectively working for Renault and Citröen, eventually created stable algorithms, using it numerically for the first time, for CAD purposes.
One issue that arises numerically is that some control points are less reliable than others, and data isn't always perfect - hence the use of what is called smoothing splines, where the curve is only required to go near enough every data point, depending on how reliable the data is and how smooth the result is to be.
The goal of this thesis is to adapt the spline method, which is used with discrete data, to continuous data, so that it is not an interpolation of a great quantity of different data points, but a smoothing of a given data function.

Chapter 2

Introduction to the Mathematical Concepts

2.1 Notation

We will first present the notations that shall be used in this thesis.
C^n is the space of continuous functions with continuous derivatives up to the order n, i.e. a C^0 function is continuous, a C^1 function is continuous and so is its derivative, and so on and so forth, until C^∞ which means the function is smooth i.e. all its derivatives are continuous.
To elaborate on that notation, we will also use $C^n(\Omega, \mathbb{R}^d)$ to denote the space of n-times continuously differentiable vector-valued functions on some open set Ω in $\mathbb{R}^d$.
We shall also use $C_0^\infty(\Omega)$ for the space of C^∞ functions on Ω that vanish identically outside of some compact $K \subset \Omega$.

2.2 Smoothing splines

We shall first explain the concept of smoothing splines. Cubic smoothing splines can be written as minimizers of the energy functional of the form:

$$J(y) = p \int_0^X y''(x)^2 dx + (1-p) \sum_{i=1}^{m} w_i (y(x_i) - y_i)^2 \tag{2.1}$$

among all y that belong to $C^2([0, X])$, with p the smoothing parameter controlling the relative importance of the smoothness of the curve in relation to the importance of the closeness of the curve to the data points, y_i the data points and w_i weights letting us adjust the importance of a point according to how reliable the data is. When p is close to 1, the cubic smoothing spline minimizer converges toward a linear function, and when p is close to 0, it resembles the least squares minimizer.

It can be proved that y the minimizing function is a third degree polynomial between the spline knots x_i, and the coefficients of the different polynomials have to match so that the curve is C^2 on the whole interval $[0, X]$ and satisfies a jump condition in the third derivative [1]. These conditions for the minimizer give rise to a linear system involving an invertible matrix, and the problem is therefore solvable with linear algebra.

Here, instead of discrete data points - the knots - which would have fit into the least square method, we are given a function f defined on subintervals of $[0, X]$, and hence it makes sense that we replace the least square part by a least square integral, and find the minimizer of that functional:

$$p \int_0^X y''(x)^2 dx + (1-p) \int_0^X w(x)(y(x) - f(x))^2 dx \tag{2.2}$$

among all y that belong to $C^2([0, X])$. $w(x)$ is a piece-wise continuous positive weight function on some interval, which describes how reliable the given data is. The minimizer to this functional is the desired curve smoothing f.
To that end we shall use calculus of variations and ODEs. Let us recall the Fundamental Lemma of Calculus of Variations from [3]:

Lemma 2.2.1 (Fundamental Lemma of Calculus of Variations) *If $h \in C^0((a, b), \mathbb{R})$ satisfies:*

$$\int_a^b h(t) \cdot \phi(t) dt = 0 \qquad \textit{for all } \phi \in C_0^\infty((a, b), \mathbb{R})$$

Then $h \equiv 0$ on (a, b).

Proof: If not, then, for some $t_0 \in (a, b)$, $h(t_0) \neq 0$. Since h is continuous, there exists some $\delta > 0$ with

$$a < t_0 - \delta < t_0 + \delta < b$$

and

$$|h(t)| > \frac{1}{2}|h(t_0)| \text{ whenever } |t_0 - t| < \delta.$$

We then choose $\phi \in C_0^\infty((a, b), \mathbb{R})$ with

$$\phi(t) = 0 \text{ if } |t_0 - t| \geq \delta$$

$$\phi(t) > 0 \text{ if } |t_0 - t| < \delta$$

For this choice of ϕ, however

$$\int_b^a h(t) \cdot \phi(t) dt = \int_{t_0-\delta}^{t_0+\delta} h(t) \cdot \phi(t) dt \neq 0,$$

contradicting our assumption. Thus, necessarily $h(t_0) = 0$ for all $t_0 \in (a, b)$.

$\square$

We shall also state a similar lemma for higher derivatives - since that will be necessary later as we're dealing with second derivatives - from [4]:

Lemma 2.2.2 (Fundamental Lemma of Calculus of Variations) *If $h_1, h_2, ..., h_n$ are continuous functions that satisfy the equality*

$$\int_a^b \phi(t) \cdot h_0(t) + \phi'(t) \cdot h_1(t) + \cdots + \phi^{(n)} \cdot h_n dt = 0 \qquad \textit{for all } \phi \in C_0^\infty((a, b), \mathbb{R})$$

Then there exist continuously differentiable functions $u_0, \ u_1, \ldots, \ u_{n-1}$ on (a, b) such that:

$$h_0 = u_0', \ h_1 = u_0 + u_1', \ldots, \ h_{n-1} = u_{n-2} + u_{n-1}', \ h_n = u_{n-1}$$

This lemma will allow us to find the desired ODE which we will then find the solution to so as to find the wanted minimizer.

In order to find the solution to said ODE, we shall also use the method of Variation of Parameters from [2]:

The method of variation of parameters for determining a particular solution of the nonhomogeneous nth order linear differential equation

$$L[y] = y^{(n)} + p_1(t)y^{(n-1)} + ... + p_{n-1}(t)y' + p_n(t)y = g(t) \tag{2.3}$$

is as follows:

It is first necessary to solve the corresponding homogeneous differential equation.
Suppose then that we know a fundamental set of solutions $y_1, y_2, ..., y_n$ of the homogeneous equation. Then the general solution of the homogeneous equation is

$$y_c(t) = c_1y_1(t) + c_2y_2(t) + ... + c_ny_n(t). \tag{2.4}$$

The method of variation of parameters for determining a particular solution of (2.3) rests on the possibility of determining n functions $u_1, u_2, ..., u_n$ such that $Y(t)$ is of the form

$$Y(t) = u_1(t)y_1(t) + u_2(t)y_2(t) + ... + u_n(t)y_n(t). \tag{2.5}$$

Since we have n functions to determine, we are free to specify n conditions. One of these should be clearly that Y satisfies (2.3). The other $n-1$ conditions are chosen so as to make the calculations as simple as possible. Since we cannot expect a simplification in determining Y if we must solve high order differential equations for $u_1, ..., u_n$, it is natural to impose conditions to suppress the terms that lead to higher derivatives of $u_1, ..., u_n$. From (2.5) we obtain

$$Y' = (u_1y_1' + u_2y_2' + ... + u_ny_n') + (u_1'y_1 + u_2'y_2 + ... + u_n'y_n), \tag{2.6}$$

where we have omitted the independent variable t on which each function in (2.6) depends. Thus the first condition that we impose is that

$$u_1'y_1 + u_2'y_2 + ... + u_n'y_n = 0. \tag{2.7}$$

Continuing this process in a similar manner through $n-1$ derivatives of Y gives

$$Y^{(m)} = u_1y_1^{(m)} + u_2y_2^{(m)} + ... + u_ny_n^{(m)}, \qquad m = 0, 1, 2, ..., n-1, \tag{2.8}$$

and the following $n-1$ conditions on the functions $u_1, ..., u_n$:

$$u_1'y_1^{(m-1)} + u_2'y_2^{(m-1)} + ... + u_n'y_n^{(m-1)} = 0, \qquad m = 1, 2, ..., n-1. \tag{2.9}$$

The nth derivative of Y is

$$Y^{(n)} = (u_1y_1^{(n)} + ... + u_ny_n^{(n)}) + (u_1'y_1^{(n-1)} + ... + u_n'y_n^{(n-1)}). \tag{2.10}$$

Finally, we impose the condition that Y must be a solution of (2.3). On substituting for the derivatives of y from (2.8) and (2.10), collecting terms, and making use of the fact that $L[y_i] = 0$, $i = 1, 2, ..., n$, we obtain

$$u_1'y_1^{(n-1)} + u_2'y_2^{(n-1)} + ... + u_n'y_n^{(n-1)} = g. \tag{2.11}$$

Equation (2.11), together with the $n-1$ equations (2.9), gives us a linear system of algebraic equations for $u_1'(t), u_2'(t), ..., u_n'(t)$:

$$\begin{cases} y_1u_1' + y_2u_2' + ... + y_nu_n' = 0 \\ y_1'u_1' + y_2'u_2' + ... + y_n'u_n' = 0 \\ y_1''u_1' + y_2''u_2' + ... + y_n''u_n' = 0 \\ \qquad\qquad\qquad\qquad \vdots \\ y_1^{(n-1)}u_1' + ... + y_n^{(n-1)}u_n' = g \end{cases} \tag{2.12}$$

The system (2.12) is a linear algebraic system for the unknown quantities $u_1'(t), ..., u_n'(t)$. By solving this system and then integrating the resulting expressions, we obtain the coefficients $u_1, ..., u_n$. A sufficient condition for the existence of the solution of the system of equations (2.12) is that the determinant of coefficients is nonzero for each value of t. However, the determinant of coefficients is the Wronskian $W(y_1, y_2, ..., y_n)$ and it is nowhere zero since $y_1, ..., y_n$ are linearly independent solutions of the homogeneous equation. Hence it is possible to determine $u_1', ..., u_n'$. We then use Cramer's rule [5]:

Theorem 2.2.3 (Cramer's Rule) *If the matrix of coefficients A is non-singular, then the unique solution $(\bar{x}_1, ..., \bar{x}_n)$ to the system*

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ \qquad\qquad\qquad\qquad\qquad\vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{cases} \tag{2.13}$$

alternatively $AX = B$, is given by

$$\bar{x}_i = (\det\ A)^{-1}\det\ (A_i) \tag{2.14}$$

where A_i is the matrix obtained from A by replacing the $i:th$ column of A by the column of constants B.

Proof: Let us use Jacobi's proof. Fix an integer h in $\{1, 2, ..., n\}$. The i-th equation of (2.13) multiplied by the cofactor A_{ih} becomes

$$A_{ih}a_{i1}x_1 + A_{ih}a_{i2}x_2 + ... + A_{ih}a_{in}x_n = A_{ih}b_i \tag{2.15}$$

Adding together the n different equations of type (2.15), we get

$$\sum_i (A_{ih}a_{i1})x_1 + ... + \sum_i (A_{ih}a_{ih})x_h + ... + \sum_i (A_{ijh}a_{in})x_n = \sum_i (A_{ih}b_i). \tag{2.16}$$

Let us then use the following theorem:

Theorem 2.2.4 *Let k and l be integers in the set $\{1, 2, ..., n\}$.*

$$\sum_{i=1}^{n} a_{il}A_{ik} = \sum_{j=1}^{n} a_{lj}A_{kj} = \begin{cases} \det\ A & if\ l = k; \text{ (First Laplace Theorem)} \\ 0 & if\ l \neq k; \text{ (Second Laplace Theorem).} \end{cases} \tag{2.17}$$

Thus, the only non-zero coefficient on the left side of (2.16) is the l-th one, and is equal to det A. On the right side we have the cofactor expansion of $\det(A_l)$ along the l-th column. Hence (2.16) reads

$$(\det\ A)x_l = \det(A_l). \tag{2.18}$$

□

Using Cramer's rule, we find that the solution of the system of equations (2.12) is

$$u_m'(t) = \frac{g(t)W_m(t)}{W(t)}, \quad m = 1, 2, ..., n. \tag{2.19}$$

Here $W(t) = W(y_1, y_2, ..., y_n)(t)$ and W_m is the determinant obtained from W by replacing the m-th column by the column $(0, 0, ..., 0, 1)$. With this notation a particular solution of Eq. (2.3) is given by

$$Y(t) = \sum_{m=1}^{n} y_m(t) \int_{t_0}^{t} \frac{g(s)W_m(s)}{W(s)} ds. \tag{2.20}$$

where t_0 is arbitrary.

Chapter 3

Curve Fitting and Minimization

3.1 Finding the ODE

As a first step of solving the general problem described in Section 2, we seek to minimize the functional:

$$J[y] = p\int_0^X y''(x)^2 dx + (1-p)\int_0^X w(x)(y(x)-f(x))^2 dx, \tag{2.2}$$

within the class of functions y that belong globally to $C^2([0,X])$ and with $w(x) \in \mathbb{R}$ piece-wise continuous, with f continuous on $[0,X]$.
Let us first demonstrate that this functional has a unique minimizer [6].

The class of functions y belong globally to $C^2([0,X])$ and f is continuous too on $[0,X]$. $C^2([0,X])$ is a linear function space and therefore convex.

We can study the convexity of the functional easily by studying the convexity of its two parts independently. A function $g : X \mapsto \mathbb{R}$ is convex if:

$$\forall x_1, x_2 \in X,\ \forall t \in [0,1]: \qquad g(tx_1 + (1-t)x_2) \leq tg(x_1) + (1-t)g(x_2), \tag{3.1}$$

and it is strictly convex if:

$$\forall x_1 \neq x_2 \in X,\ \forall t \in [0,1]: \qquad g(tx_1 + (1-t)x_2) < tg(x_1) + (1-t)g(x_2). \tag{3.2}$$

Let us first study the convexity of the first part:

$$p\int_0^X y''(x)^2 dx,$$

where we know that $x \mapsto x^2$ is strictly convex. Therefore:

$$p\int_0^X (ty_1''(x) + (1-t)y_2''(x))^2 dx \leq p\int_0^X t(y_1''(x))^2 + (1-t)(y_2''(x))^2 dx$$

and thus this first part is clearly convex. It is not necessarily strictly convex, however, as equality, which simplifies to $y_1'' = y_2''$, is possible even with $y_1 \neq y_2$ (i.e. $y_1 = y_2 + Ax + B$ for some $A, B \in \mathbb{R}$.)
Let us now study the second part:

$$(1-p)\int_0^X w(x)(y(x)-f(x))^2 dx,$$

where, same as before, we know that $x \mapsto x^2$ is strictly convex. Therefore:

$$(1-p)\int_0^X w(x)\big(ty_1(x)+(1-t)y_2(x)-f(x)\big)^2dx = (1-p)\int_0^X w(x)\big(t(y_1(x)-f(x))+(1-t)(y_2(x)-f(x))\big)^2dx$$
$$\leq (1-p)\int_0^X w(x)t(y_1(x)-f(x))^2 + w(x)(1-t)(y_2(x)-f(x))^2dx,$$

and this is therefore also convex. It is not, however, strictly convex, as $w(x) = 0$ clearly leads to an equality, although it is indeed strictly convex on intervals where $w(x) \neq 0$, as equality, which simplifies to $y_1 = y_2$, would imply the falsity of the initial hypothesis of $y_1 \neq y_2$.
We therefore have that the functional $J[y]$ is convex and strictly convex on intervals (here we only have one) where $w(x) \neq 0$. From that, we can deduce that a minimizer always exists, and that it is unique when $w(x)$ is not identically 0.

□

We then assume y_* is a minimum function for y. Let $y(x) = y_*(x) + \varepsilon\eta(x)$, $\varepsilon \in \mathbb{R}$ a constant and η an arbitrary fixed C^2 function defined on $[0, X]$. The term $\varepsilon\eta(x)$ forms the variation.

We replace $y(x)$ with $y_*(x) + \varepsilon\eta(x)$ and rewrite (2.2) as a function of ε as:

$$I(\varepsilon) = p\int_0^X (y_*''(x)+\varepsilon\eta''(x))^2dx + (1-p)\int_0^X w(x)(y_*(x)+\varepsilon\eta(x)-f(x))^2dx, \tag{3.3}$$

It is clear that $I(\varepsilon)$ has minimum at $\varepsilon = 0$ as this gives $y(x) = y_*(x)$, the minimum function; hence $I'(0) = 0$. By expanding the squares and differentiating with relation to ε we get:

$$I'(0) = \int_0^X p\, y_*''(x)\eta''(x) + (1-p)w(x)\eta(x)(y_*(x)-f(x))\; dx = 0, \tag{3.4}$$

which we can rewrite, so as to use the Fundamental Lemma of Calculus of Variations (2.2.2) for higher derivatives, as:

$$I'(0) = \int_0^X \eta''(x)h_2(x) + \eta(x)h_0(x)\; dx = 0. \tag{3.5}$$

We can then apply the Fundamental Lemma since η is a compactly supported smooth function on $[0, X]$. This gives us the following system:

$$\begin{cases} h_0(x) = (1-p)w(x)(y_* - f(x)) & = u_0'(x) \\ h_1(x) = 0 & = u_0(x) + u_1'(x) \\ h_2(x) = py_*''(x) & = u_1(x) \end{cases}$$

and the Fundamental Lemma of Calculus of Variations says those u_0 and u_1 are continuously differentiable. This also implies u_1' is continuously differentiable since $u_1'(x) = -u_0(x)$ and so we can write:

$$u_1'(x) + u_0(x) = 0 \implies u_1''(x) + u_0'(x) = 0.$$

By replacing u_0 and u_1 we obtain the following ODE:

$$p\,y_*^{(4)}(x) + (1-p)w(x)(y_*(x)-f(x)) = 0, \tag{3.6}$$

or alternatively written as:

$$y_*^{(4)}(x) + \left(\frac{1-p}{p}\right) w(x)y_*(x) = \left(\frac{1-p}{p}\right) w(x)f(x). \tag{3.7}$$

This gives us, because of the conditions on u_0 and u_1 given to us by the Fundamental Lemma of Calculus of Variations, that the class of functions y are C^4. This fourth-order ODE gives us the minimizations of the functional for $w(x)$ constant on one interval - we shall study later what happens with different $w(x)$ i.e. different subintervals.

3.2 Finding the solution to the ODE

We assume that $w(x) \in \{0,1\}$ constant on an interval. We first solve the case when $w(x) = 0$, which gives us a third degree polynomial:

$$y_*(x) = c_1x^3 + c_2x^2 + c_3x + c_4 \tag{3.8}$$

We will now study the ODE if $w(x) = 1$. (3.6) can be rewritten as:

$$p\, y_*^{(4)}(x) + (1-p)y_*(x) = (1-p)f(x) \tag{3.9}$$

So as to solve the fourth degree ODE, we use the Variation of Parameters method we described in Chapter (2).

We first solve the ODE in the homogeneous case $f(x) = 0$, which gives us:

$$y_h(x) = c_1e^{\alpha(1+i)x} + c_2e^{-\alpha(1+i)x} + c_3e^{\alpha(1-i)x} + c_4e^{-\alpha(1-i)x}, \tag{3.10}$$

with $\alpha = \frac{\sqrt{2}\sqrt[4]{\frac{(1-p)}{p}}}{2}$.
We can alternatively write this homogeneous solution as:

$$y_h(x) = c_1\cosh(\alpha x)\cos(\alpha x) + c_2\cosh(\alpha x)\sin(\alpha x) + c_3\sinh(\alpha x)\cos(\alpha x) + c_4\sinh(\alpha x)\sin(\alpha x). \tag{3.11}$$

To find a particular solution to the the nonhomogeneous equation we look for a solution of the form:

$$y_* = u_1y_1 + u_2y_2 + u_3y_3 + u_4y_4 \tag{3.12}$$

u_1, u_2, u_3, u_4 are unknown functions and

$$\begin{cases} y_1(x) = \cosh(\alpha x)\cos(\alpha x), \\ y_2(x) = \cosh(\alpha x)\sin(\alpha x), \\ y_3(x) = \sinh(\alpha x)\cos(\alpha x), \\ y_4(x) = \sinh(\alpha x)\sin(\alpha x). \end{cases}$$

We take for $n \in \{1,2,3,4\}$, u_n such that $u_1'y_1^{m-1} + u_2'y_2^{m-1} + u_3'y_3^{m-1} + u_4'y_4^{m-1} = 0$ for $m \in \{1,2,3\}$. This gives us $u_1'y_1''' + u_2'y_2''' + u_3'y_3''' + u_4'y_4''' = f$ and therefore the following system of equations:

$$\begin{cases} u_1'y_1 + u_2'y_2 + u_3'y_3 + u_4'y_4 = 0 \\ u_1'y_1' + u_2'y_2' + u_3'y_3' + u_4'y_4' = 0 \\ u_1'y_1'' + u_2'y_2'' + u_3'y_3'' + u_4'y_4'' = 0 \\ u_1'y_1''' + u_2'y_2''' + u_3'y_3''' + u_4'y_4''' = f. \end{cases} \tag{3.13}$$

By using Cramer's Rule (2.2.3), this simplifies to:

$$\begin{aligned} y_*(x) = \left(\frac{1-p}{p}\right) & \left[y_1(x)\int_0^x \frac{f(\chi)W_1(\chi)}{W(\chi)}d\chi + y_2(x)\int_0^x \frac{f(\chi)W_2(\chi)}{W(\chi)}d\chi \right. \\ & \left. + y_3(x)\int_0^x \frac{f(\chi)W_3(\chi)}{W(\chi)}d\chi + y_4(x)\int_0^x \frac{f(\chi)W_4(\chi)}{W(\chi)}d\chi \right], \end{aligned} \tag{3.14}$$

where W is the determinant of the Wronskian for the system and W_i ($i \in \{1,2,3,4\}$) is the determinant for the same Wronskian with the i-th column replaced by $(0,0,0,1)$.

To obtain an explicit formula we need to compute the Wronskian of the fundamental system of solutions, which shall then be used to find the W and W_i of the above equation:

$$W = \begin{vmatrix} y_1 & y_2 & y_3 & y_4 \\ y_1' & y_2' & y_3' & y_4' \\ y_1'' & y_2'' & y_3'' & y_4'' \\ y_1''' & y_2''' & y_3''' & y_4''' \end{vmatrix} \tag{3.15}$$

where

$$\begin{aligned} y_1(x) &= \cosh(\alpha x)\cos(\alpha x) \\ y_2(x) &= \cosh(\alpha x)\sin(\alpha x) \\ y_3(x) &= \sinh(\alpha x)\cos(\alpha x) \\ y_4(x) &= \sinh(\alpha x)\sin(\alpha x) \end{aligned}$$

The W_i are of the form:

$$W_1 = \begin{vmatrix} 0 & y_2 & y_3 & y_4 \\ 0 & y_2' & y_3' & y_4' \\ 0 & y_2'' & y_3'' & y_4'' \\ 1 & y_2''' & y_3''' & y_4''' \end{vmatrix} \quad W_2 = \begin{vmatrix} y_1 & 0 & y_3 & y_4 \\ y_1' & 0 & y_3' & y_4' \\ y_1'' & 0 & y_3'' & y_4'' \\ y_1''' & 1 & y_3''' & y_4''' \end{vmatrix} \quad W_3 = \begin{vmatrix} y_1 & y_2 & 0 & y_4 \\ y_1' & y_2' & 0 & y_4' \\ y_1'' & y_2'' & 0 & y_4'' \\ y_1''' & y_2''' & 1 & y_4''' \end{vmatrix} \quad W_4 = \begin{vmatrix} y_1 & y_2 & y_3 & 0 \\ y_1' & y_2' & y_3' & 0 \\ y_1'' & y_2'' & y_3'' & 0 \\ y_1''' & y_2''' & y_3''' & 1 \end{vmatrix}$$

After heavy symbolic computations on MATLAB, we obtain the solutions to the nonhomogeneous system:

$$\begin{aligned} y_*(x) = y_h(x) + \left(\frac{1-p}{\beta p}\right) \Bigg[&-\cosh(\alpha x)\cos(\alpha x)\int_0^x (\cosh(\alpha\chi)\sin(\alpha\chi) - \sinh(\alpha\chi)\cos(\alpha\chi))f(\chi)d\chi \\ &+\cosh(\alpha x)\sin(\alpha x)\int_0^x (\cosh(\alpha\chi)\cos(\alpha\chi) + \sinh(\alpha\chi)\sin(\alpha\chi))f(\chi)d\chi \\ &-\sinh(\alpha x)\cos(\alpha x)\int_0^x (\cosh(\alpha\chi)\cos(\alpha\chi) - \sinh(\alpha\chi)\sin(\alpha\chi))f(\chi)d\chi \\ &-\sinh(\alpha x)\sin(\alpha x)\int_0^x (\cosh(\alpha\chi)\sin(\alpha\chi) + \sinh(\alpha\chi)\cos(\alpha\chi))f(\chi)d\chi \Bigg], \end{aligned} \tag{3.16}$$

where $y_h(x) = c_1\cosh(\alpha x)\cos(\alpha x) + c_2\cosh(\alpha x)\sin(\alpha x) + c_3\sinh(\alpha x)\cos(\alpha x) + c_4\sinh(\alpha x)\sin(\alpha x)$, $\alpha = \frac{\sqrt{2}\sqrt[4]{\frac{(1-p)}{p}}}{2}$ and $\beta = \sqrt{2}\left(\frac{1-p}{p}\right)^{(3/4)}$.

We alternatively, for convenience, write this equation as:

$$y_*(x) = y_h(x) + y_p(x) \tag{3.17}$$

if $w(x) = 1$ and

$$y_*(x) = c_1x^3 + c_2x^2 + c_3x + c_4 \tag{3.18}$$

if $w(x) = 0$, which, in the case of different subintervals in the future, will lead to different c_1, c_2, c_3 and c_4 for every subinterval and different $w(x)$.

We now stop assuming that $w(x)$ is constant on the entire interval, but only piece-wise constant, such that we have subintervals with different values of $w(x)$. We therefore need to study what happens at boundaries.

3.3 Boundary Conditions

We have studied what happens for the minimizer symbolically generally, but we need, so as to find the coefficients of the minimizer and therefore the minimizer of the variational problem

$$J[y] = p\int_0^X y''(x)^2 dx + (1-p)\int_0^X w(x)(y(x)-f(x))^2 dx, \tag{2.2}$$

to have some boundary conditions on the interval $[0, X]$ with $w(x)$ constant on the interval. With a given data function we will have some boundary conditions (depending on the context) and we need to find ways to link them to our variational problem. We first study this problem again, so as to find boundary conditions belonging to this minimization problem.
We know, from the Fundamental Lemma of Calculus of Variations that we used to find the fourth-order ODE, that y_* is C^4 on $[0, X]$. We can now use that knowledge to compute the constraints on the boundaries stemming from the integral. First off, we integrate $\int_0^X 2\ p\ y_*''(x)\eta''(x)dx$ by parts and we get:

$$I'(0) = \int_0^X \eta(x)\Big(p\, y_*^{(4)}(x) + w(x)(1-p)(y_*(x)-f(x))\Big)\, dx + p\Big(\big[y_*''(x)\eta'(x)\big]_0^X - \big[y_*'''(x)\eta(x)\big]_0^X\Big) = 0, \tag{3.19}$$

and let $\Theta = p\Big(\big[y_*''(x)\eta'(x)\big]_0^X - \big[y_*'''(x)\eta(x)\big]_0^X\Big)$.
If we take $\eta(x)$ such that it and its derivative vanish at the boundaries, then $\Theta = 0$ and we can then apply the Fundamental Lemma of Calculus of Variations (2.2.1), which tells us that, since all η are smooth and compact on $[0, X]$, the rest is equivalently equal to 0, which gives us the same fourth-order ODE as before.

We shall then investigate for different boundary conditions.

3.3.1 Fixed Boundary Conditions

For fixed boundary conditions:

$$\begin{cases} y_*(0) = a \\ y_*(X) = b \\ y_*'(0) = 0 \\ y_*'(X) = 0. \end{cases} \tag{3.20}$$

This gives $(c_1, c_2, c_3, c_4) = (2\dfrac{a-b}{X^3}, 3\dfrac{b-a}{X^2}, 0, a)$ for $w(x) = 0$.
The minimizer then is:

$$y_*(x) = 2\frac{a-b}{X^3}x^3 + 3\frac{b-a}{X^2}x^2 + a. \tag{3.21}$$

For $w(x) = 1$ we have:

$$\begin{cases} y_*(0) = y_h(0) + y_p(0) = c_1 + c_3 = a \\ y_*(X) = y_h(X) + y_p(X) = b \\ y_*'(0) = y_h'(0) + y_p'(0) = \frac{\alpha}{2}(c_1 + c_2 - c_3 + c_4) = 0 \\ y_*'(X) = y_h'(X) + y_p'(X) = 0, \end{cases} \tag{3.22}$$

where computations give $y_p(0) = y_p'(0) = 0$.
(The minimizer given is too long to be written here.)

3.3.2 Free Boundary Conditions

For free boundary conditions, that is if we have an unconstrained variational problem, we need to find conditions depending directly on the ODE, i.e. on Θ here. We first fix different η and relate it to Θ:
We take $\eta(0) = \eta(X) = \eta'(0) = \eta'(X) = 0$ except at one of those boundaries at a time:

$$\Theta = 2p\Big(\big[y_*''(x)\eta'(x)\big]_0^X - \big[y_*'''(x)\eta(x)\big]_0^X\Big) = \begin{cases} 2p\,y_*'''(0) = 0 & \text{for } \eta(0) = 1 \\ -2p\,y_*'''(X) = 0 & \text{for } \eta(X) = 1 \\ -2p\,y_*''(0) = 0 & \text{for } \eta'(0) = 1 \\ 2p\,y_*''(X) = 0 & \text{for } \eta'(X) = 1, \end{cases} \tag{3.23}$$

which gives the system corresponding to free boundary conditions

$$\begin{cases} y_*'''(X) = 0 \\ y_*'''(0) = 0 \\ y_*''(X) = 0 \\ y_*''(0) = 0, \end{cases} \tag{3.24}$$

which, with $w(x) = 0$, gives $c_1 = c_2 = 0$ with c_3 and c_4 free, that is:

$$y_*(x) = c_3 x + c_4. \tag{3.25}$$

We therefore don't have uniqueness of the minimizer.
The system for $w(x) = 1$ is too complicated to write down here.

3.3.3 Periodic Boundary Conditions

For periodic boundary conditions (that is, ${y_*}^{(m)}(0) = {y_*}^{(m)}(X)$ for $m \in \{0, 1, 2, 3\}$), we get:

$$\begin{cases} y_*(0) = y_*(X) \\ y_*'(0) = y_*'(X) \\ y_*''(0) = y_*''(X) \\ y_*'''(0) = y_*'''(X) \end{cases} \tag{3.26}$$

which, in the case $w(x) = 0$, gives $c_1 = c_2 = c_3 = 0$, with c_4 free (i.e. y_* is a constant function.) We therefore don't have uniqueness of the minimizer.
The system for $w(x) = 1$ is too complicated to write down here.

3.3.4 Fitting Subintervals

We can now start looking at cases where the weight function varies on different intervals. We know, from the lemma of calculus of variations, that on any subinterval where $w(x)$ is constant, y_* satisfies the ODE. For some boundary conditions, we study Θ for the general case with $n+1$ subintervals $[x_i, x_{i+1}]$, with $w(x)_{[x_i,x_{i+1}]} \neq w(x)_{[x_{i+1},x_{i+2}]}$:

$$2\int_0^X \eta(x)\Big(p\,y_*^{(4)}(x) + w(x)(1-p)(y_*(x)-f(x))\Big)\,dx + \Theta = 0$$

thus, equivalently, for $n+1$ subintervals, this gives:

$$\sum_{i=0}^{n} 2\int_{x_i}^{x_{i+1}} \eta(x)\Big(p\,y_*^{(4)}(x) + w(x)(1-p)(y_*(x)-f(x))\Big)\,dx + \Theta_{[x_i,x_{i+1}]} = 0$$

then, for any $i \in \mathbb{N}$ between 1 and $n-1$, this gives:

$$\begin{aligned}&2\int_{x_{i-1}}^{x_i} \eta(x)\Big(p\,y_*^{(4)}(x) + w(x)(1-p)(y_*(x)-f(x))\Big)\,dx +\\ &2\int_{x_i}^{x_{i+1}} \eta(x)\Big(p\,y_*^{(4)}(x) + w(x)(1-p)(y_*(x)-f(x))\Big)\,dx + \Theta_{[x_{i-1},x_i]} + \Theta_{[x_i,x_{i+1}]} = 0\end{aligned} \tag{3.27}$$

where $\Theta_{[x_{i-1},x_1]} + \Theta_{[x_i,x_{i+1}]} = 0$ which gives

$$y_*''(x_{i-})\eta'(x_{i-}) - y_*'''(x_{i-})\eta(x_{i-}) = y_*''(x_{i+})\eta'(x_{i+}) - y_*'''(x_{i+})\eta(x_{i+})$$

where x_{i-} is the neighbourhood of x_i below, and x_{i+} is the neighbourhood of x_i above.
We also want the function to be continuous and differentiable, as y belongs to $C^2[0,X]$, hence $y_*(x_{i-}) = y_*(x_{i+})$ and $y_*'(x_{i-}) = y_*'(x_{i+})$. This gives us the system:

$$\begin{cases} y_*(x_{i-}) = y_*(x_{i+}) \\ y_*'(x_{i-}) = y_*'(x_{i+}) \\ y_*''(x_{i-}) = y_*''(x_{i+}) \\ y_*'''(x_{i-}) = y_*'''(x_{i+}) \end{cases} \tag{3.28}$$

which can then be repeated for every neighbourhood for every two adjacent subintervals.

So, with a known amount n of subintervals, we can write the system so as to find the n different parameters. We first start off with the choice of fixed, free or periodic boundaries, and to this we append the internal systems between each subinterval, x_i being the $n-1$ values where two subintervals meet:

$$\begin{cases} y_*(x_{1-}) = y_*(x_{1+}) \\ y_*'(x_{1-}) = y_*'(x_{1+}) \\ y_*''(x_{1-}) = y_*''(x_{1+}) \\ y_*'''(x_{1-}) = y_*'''(x_{1+}) \\ y_*(x_{2-}) = y_*(x_{2+}) \\ y_*'(x_{2-}) = y_*'(x_{2+}) \\ y_*''(x_{2-}) = y_*''(x_{2+}) \\ y_*'''(x_{2-}) = y_*'''(x_{2+}) \\ \qquad\vdots \\ y_*(x_{n-1_-}) = y_*(x_{n-1_+}) \\ y_*'(x_{n-1_-}) = y_*'(x_{n-1_+}) \\ y_*''(x_{n-1_-}) = y_*''(x_{n-1_+}) \\ y_*'''(x_{n-1_-}) = y_*'''(x_{n-1_+}) \end{cases} \tag{3.29}$$

Solving this system with its corresponding boundary conditions system appended will in turn give us the matrix of all n coefficients of the form:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & c_{n3} & c_{n4}. \end{bmatrix} \tag{3.30}$$

We therefore have an approach so as to compute the minimizer, we now have to figure out how to approach this numerically.

Chapter 4

Numerical Methods

Now that we have the theory behind computing the minimizer, we need to find numerical methods to compute it, keeping in mind both efficiency and accuracy.
There are two ways to go about to doing this: one way would be to solve the ODE numerically directly, another, which we shall do below, is to implement the results we obtained from solving the ODE symbolically above.
The equation (3.16), the solution when $w(x) = 1$, contains four integrals, to which the primitives are indefinite, which is explained by the fact we will not use an actual function $f(x)$, contrary to what we assumed until now, but instead an array of data points sufficiently close so that they might be considered continuous, as this is how data is, most often, stored. We must thus first find a way to compute those numerically [7].

4.1 Numerical Integration

4.1.1 Trapezoidal Rule

The first method that makes sense to use, as it is a very simple and fast one, is the trapezoidal rule, alternatively the first degree Newton-Cotes formula.
The trapezoidal rule is a method to evaluate the definite integral

$$\int_a^b f(x)dx.$$

It works by approximating regions under the integral as the area of trapezoids from a point $(a, f(a))$ to another $(b, f(b))$, i.e., for a single area:

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}.$$

It makes sense then to use a uniform grid with n points to approximate the integral and, the smaller the step $\Delta x = \frac{b-a}{n}$, the more accurate the approximation:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}\sum_{i=1}^{n}(f(x_{i-1}) + f(x_i)). \tag{4.1}$$

The error induced by the approximation is the difference between the integral and its approximation by the trapezoidal rule

$$\text{error} = \int_a^b f(x)dx - \Delta x\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(a + i\Delta x)\right]. \tag{4.2}$$

Theorem 4.1.1 *Assuming f is C^2, then there exists a number ξ between a and b such that*

$$\text{error} = -\frac{(b-a)^3}{12n^2} f''(\xi). \tag{4.3}$$

Proof: First suppose that $h = \frac{b-a}{n}$, $a_k = a + (k-1)h$, and let $g_k(t) = \frac{1}{2}t[f(a_k) + f(a_k + t)] - \int_{a_k}^{a_k+t} f(x)dx$ be the function such that $|g_k(h)|$ is the error of the trapezoidal rule on one of the intervals $[a_k, a_{k+h}]$. Then:

$$\begin{aligned}
\frac{dg_k}{dt} &= \frac{1}{2}[f(a_k) + f(a_k + t)] + \frac{1}{2}tf'(a_k + t) - f(a_k + t), \\
\frac{d^2 g_k}{dt^2} &= \frac{1}{2}tf''(a_k + t).
\end{aligned}$$

Suppose that $|f''(x)| \leq f''(\xi)$, so that:

$$|f''(a_k + t)| \leq f''(\xi), \; -f''(\xi) \leq f''(a_k + t) \leq f''(\xi),$$
$$-\frac{f''(\xi)t}{2} \leq g_k''(t) \leq \frac{f''(\xi)t}{2}.$$

Since $g_k'(0) = 0$ and $g_k(0) = 0$, we have

$$\int_0^t g_k''(x)dx = g_k'(t)$$
$$\int_0^t g_k'(x)dx = g_k(t),$$

so that:

$$-\frac{f''(\xi)t^2}{4} \leq g_k'(t) \leq \frac{f''(\xi)t^2}{4}$$
$$-\frac{f''(\xi)t^3}{12} \leq g_k(t) \leq \frac{f''(\xi)t^3}{12}$$

We then replace t by h:

$$-\frac{f''(\xi)h^3}{12} \leq g_k(h) \leq \frac{f''(\xi)h^3}{12}.$$

And

$$\sum_{k=1}^{n} g_k(h) = \frac{b-a}{n}\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a + k\frac{b-a}{n})\right] - \int_a^b f(x)dx,$$
$$\sum_{k=1}^{n} \frac{f''(\xi)h^3}{12} = \frac{f''(\xi)h^3 n}{12}.$$

So:

$$-\frac{f''(\xi)h^3 n}{12} \leq \frac{b-a}{n}\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a + k\frac{b-a}{n})\right] - \int_a^b f(x)dx \leq \frac{f''(\xi)h^3 n}{12},$$
$$\text{error} = \int_a^b f(x)dx - \frac{b-a}{n}\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a + k\frac{b-a}{n})\right] = \frac{f''(\xi)h^3 n}{12} = \frac{f''(\xi)(b-a)^3}{12n^2}.$$

□

We also deduce that the trapezoidal rule is $O(h^2)$.
We can easily implement this method in MATLAB so as to compute the four integrals needed to get the minimizer.
Since the trapezoidal rule is a second-order method for numerical integration, it is not the most accurate and thus the error can lead to issues in the long run. Other methods exist and can be implemented, such as Simpson's rule.

4.1.2 Simpson's Rule

The Simpson's rule, alternatively the second degree Newton-Cotes formula, is a method for numerical integration which uses quadratic interpolation. It replaces the integrand $f(x)$ with a quadratic interpolating polynomial $p_2(f)$ on $[a,b]$. We take $c = (a+b)/2$ and get

$$\int_a^b f(x)dx \approx \int_a^b \Big[\frac{(x-c)(x-b)}{(a-c)(a-b)}f(a) + \frac{(x-a)(x-b)}{(c-a)(c-b)}f(c) + \frac{(x-a)(x-c)}{(b-a)(b-c)}f(b)\Big]dx.$$

Integrating this, we obtain

$$\int_a^b f(x)dx \approx \frac{h}{3}\big[f(a) + 4f\Big(\frac{a+b}{2}\Big) + f(b)\big], \quad h = \frac{b-a}{2}. \tag{4.4}$$

The error induced by the method, with a number ξ between a and b, is

$$\text{error} = -\frac{h^5}{90}f^{(4)}(\xi). \tag{4.5}$$

4.2 Implementation

We now have several ways to compute the integrals of

$$\begin{aligned} y_*(x) = y_h(x) + \left(\frac{1-p}{\beta p}\right) \Bigg[&-\cosh(\alpha x)\cos(\alpha x)\int_0^x (\cosh(\alpha\chi)\sin(\alpha\chi) - \sinh(\alpha\chi)\cos(\alpha\chi))f(\chi)d\chi \\ &+\cosh(\alpha x)\sin(\alpha x)\int_0^x (\cosh(\alpha\chi)\cos(\alpha\chi) + \sinh(\alpha\chi)\sin(\alpha\chi))f(\chi)d\chi \\ &-\sinh(\alpha x)\cos(\alpha x)\int_0^x (\cosh(\alpha\chi)\cos(\alpha\chi) - \sinh(\alpha\chi)\sin(\alpha\chi))f(\chi)d\chi \\ &-\sinh(\alpha x)\sin(\alpha x)\int_0^x (\cosh(\alpha\chi)\sin(\alpha\chi) + \sinh(\alpha\chi)\cos(\alpha\chi))f(\chi)d\chi\Bigg]. \end{aligned} \tag{3.16}$$

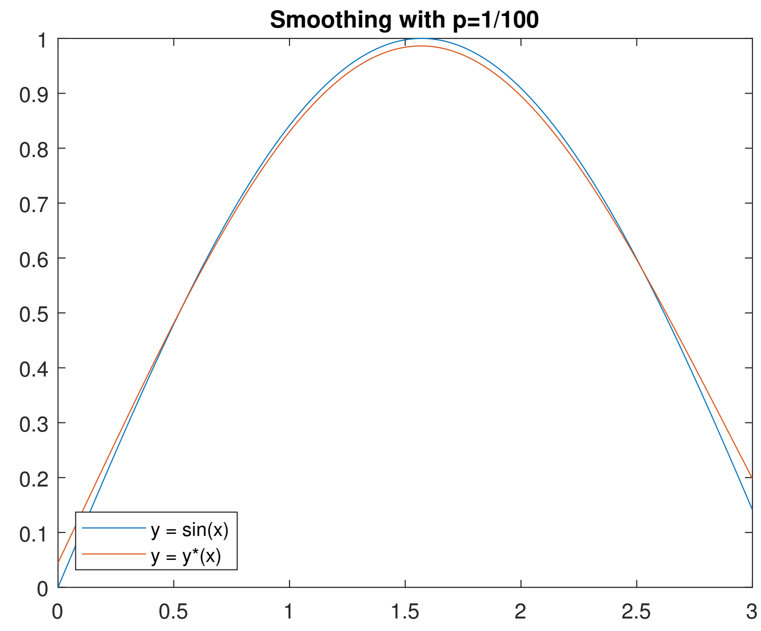
We now wish to create a method that solves the minimization problem, such that it can use several subintervals where $w(x) = 1$ - inducing the above solution - and subintervals where $w(x) = 0$, inducing

$$y_*(x) = c_1x^3 + c_2x^2 + c_3x + c_4. \tag{3.8}$$

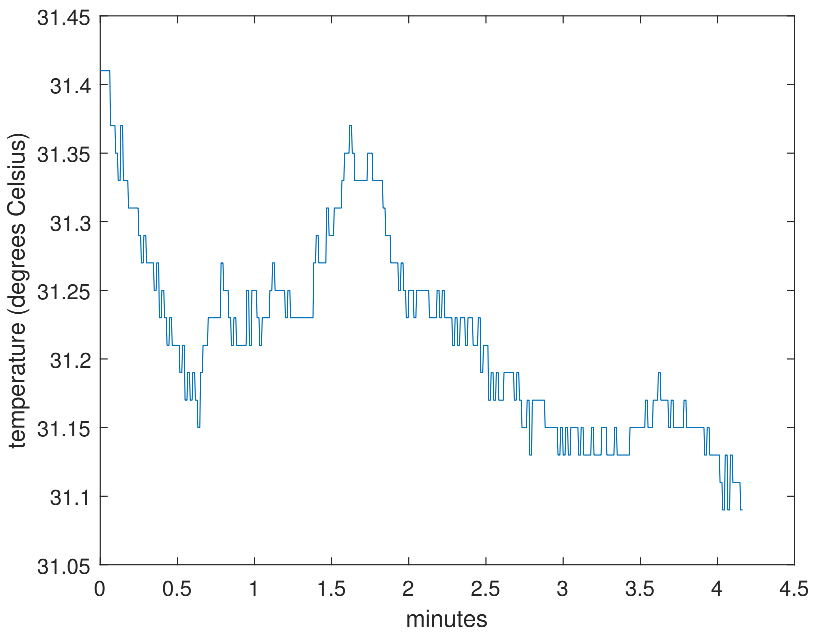
In order to do that we implement the system of equations we put forward before in (3.29), and let us adjoin it, for example, free boundary conditions (3.20):

$$\begin{cases} y_*''(0) & = 0 \\ y_*'''(0) & = 0 \\ y_*(x_{1-}) & = y_*(x_{1+}) \\ {y_*}'(x_{1-}) & = {y_*}'(x_{1+}) \\ {y_*}''(x_{1-}) & = {y_*}''(x_{1+}) \\ {y_*}'''(x_{1-}) & = {y_*}'''(x_{1+}) \\ y_*(x_{2-}) & = y_*(x_{2+}) \\ {y_*}'(x_{2-}) & = {y_*}'(x_{2+}) \\ {y_*}''(x_{2-}) & = {y_*}''(x_{2+}) \\ {y_*}'''(x_{2-}) & = {y_*}'''(x_{2+}) \\ \qquad\vdots & \\ y_*(x_{n-1_-}) & = y_*(x_{n-1_+}) \\ {y_*}'(x_{n-1_-}) & = {y_*}'(x_{n-1_+}) \\ {y_*}''(x_{n-1_-}) & = {y_*}''(x_{n-1_+}) \\ {y_*}'''(x_{n-1_-}) & = {y_*}'''(x_{n-1_+}) \\ y_*''(X) & = 0 \\ y_*'''(X) & = 0. \end{cases} \tag{4.6}$$

We implement this system of equations above symbolically, and we then find solutions numerically to this system, using the integrals we computed with the trapezoidal rule. We can then compute the minimizer. So as to give an example we will need a data set. We can easily do a first try with a simple smooth function such as $\sin(x)$ with free boundary conditions:

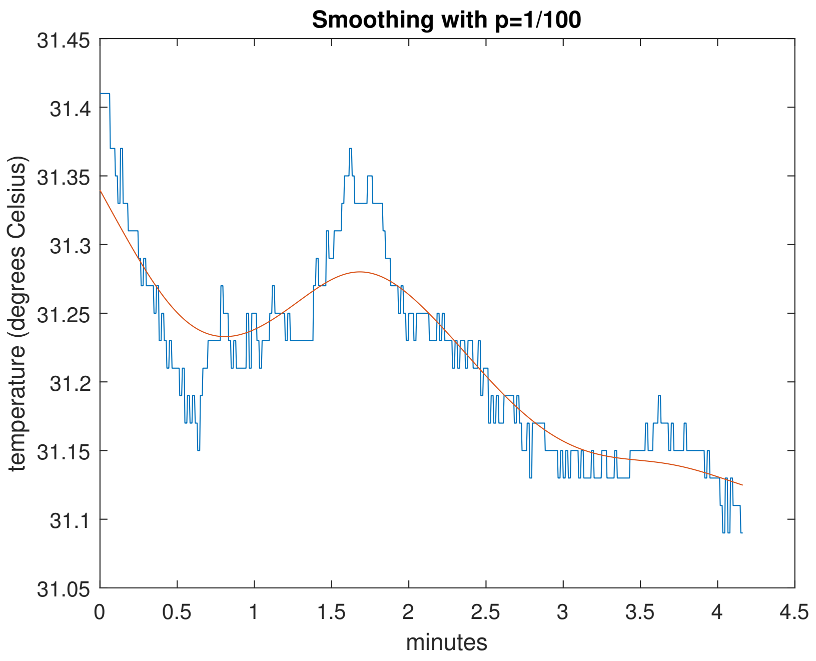


Where we can clearly see the two curves being close to each other, with the minimizer following closely, on the set of a 1000 data points from 0 to 3 uniformously distributed.
Let us use now use, for a more concrete and relevant example, with a non-smooth data function,the WESAD [10] data set as an example. Here we will take the measurements of outside body temperature at the wrist of a man. The first thousand points of data are such:



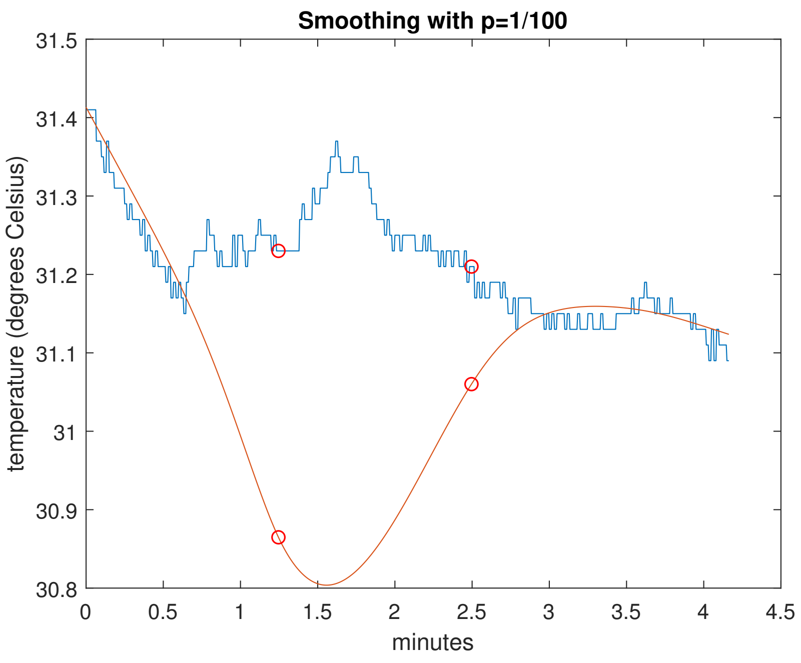
Where clearly smoothing would be useful.

We can then execute our algorithm, with say, the smoothing parameter $p = 1/100$ and using the trapezoidal rule:

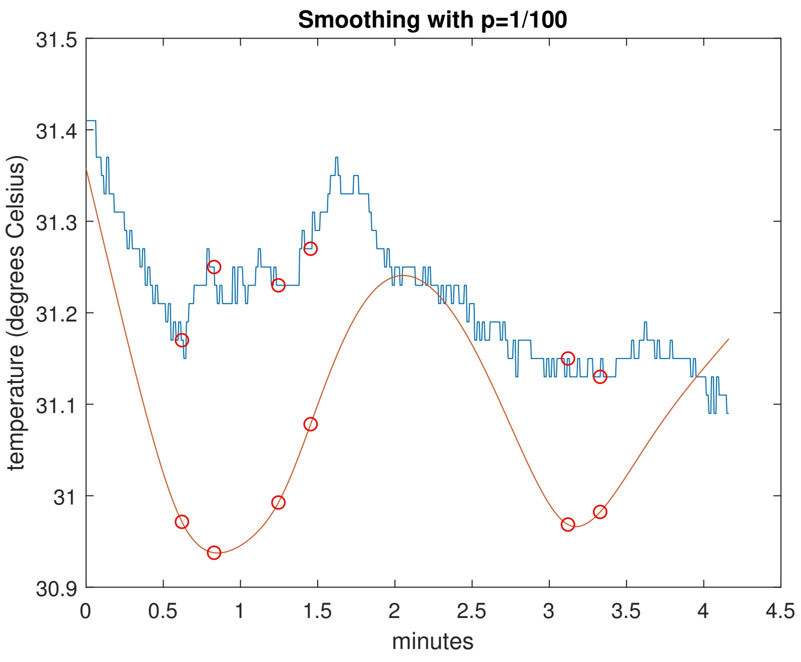


Which clearly smoothes the data and thus illustrates that the method does what it is supposed to do.

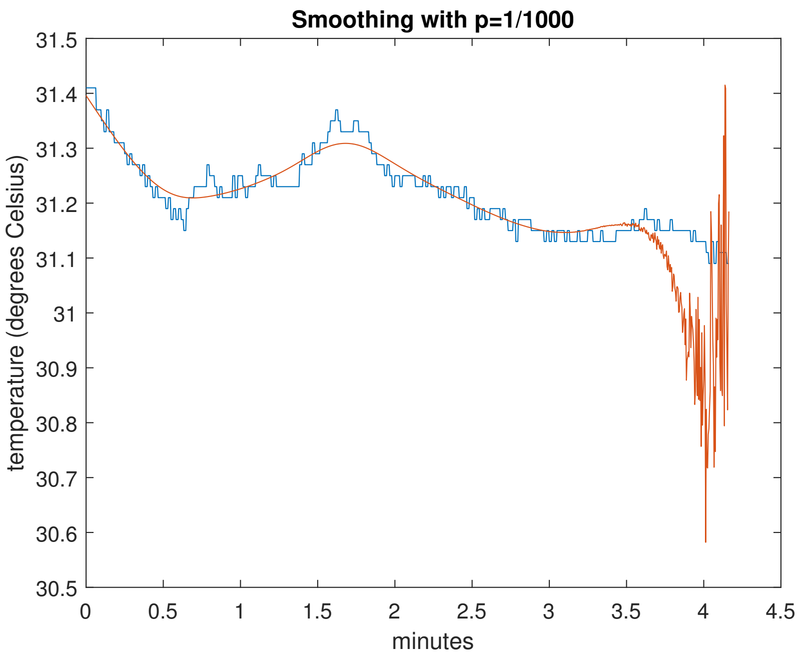
We can then try by imagining we don't know the data between two points, so that it would be an interval with $w(x) = 0$, say between the 300th and 600th point (i.e. between ≈ 1.25 minutes and ≈ 2.5 minutes):



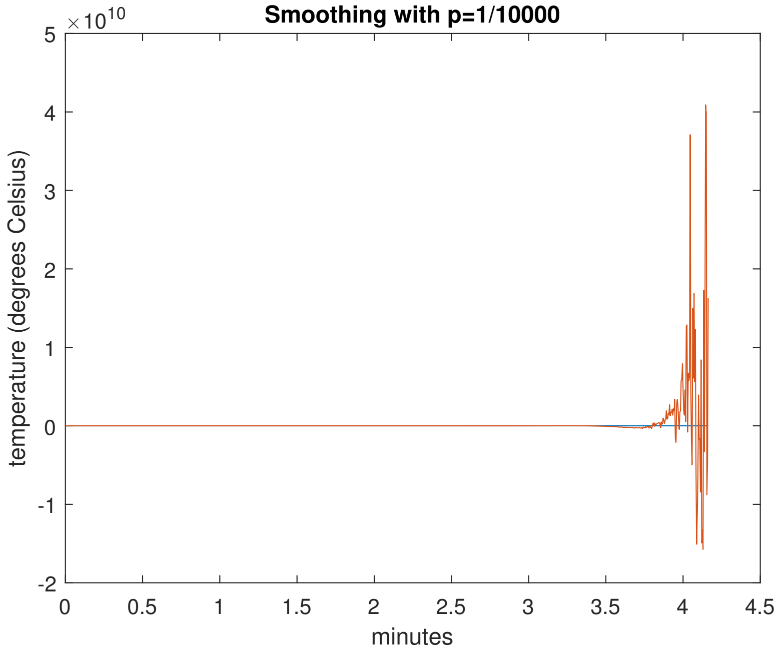
We can also show it easily keeps the result smooth even for more than two subintervals, say we don't know the data between the 150th and 200th point, the 300th and the 350th point, and the 750th and 800th point (respectively ≈ 0.62 minutes, ≈ 0.82 minutes, ≈ 1.25 minutes, ≈ 1.45 minutes, ≈ 3.12 minutes, and ≈ 3.32 minutes):



There are, however, issues with this method, which are due to the integrals of 3.16 containing exponential functions of αx, hence, for an αx high enough (if we increase x and/or lower p), the solution starts to diverge and completely, after a while, becomes too big to be even computed by MATLAB.
If we were to take, for the same interval, $p = 1/1000$ (and $w(x) = 1$ on the entire interval):

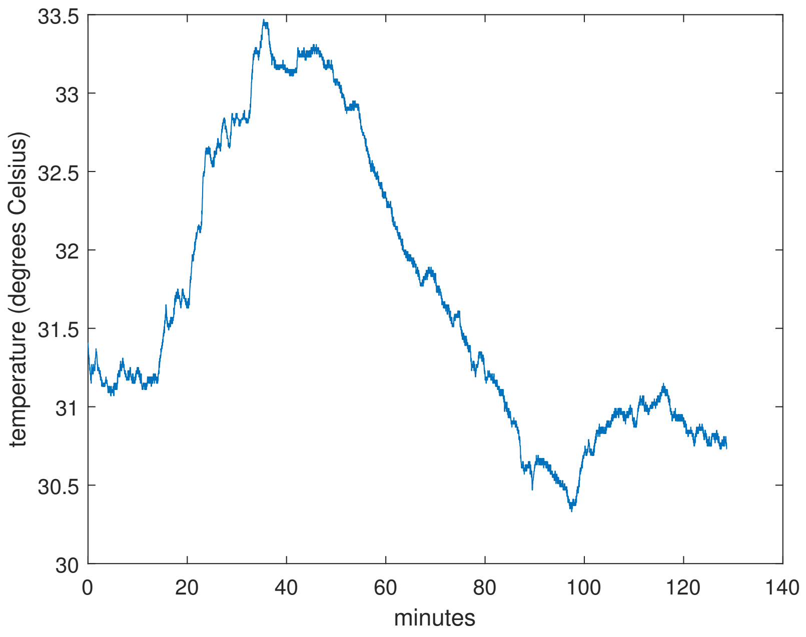


Which clearly diverges at the end due to an αx too high. For even smaller p we get:

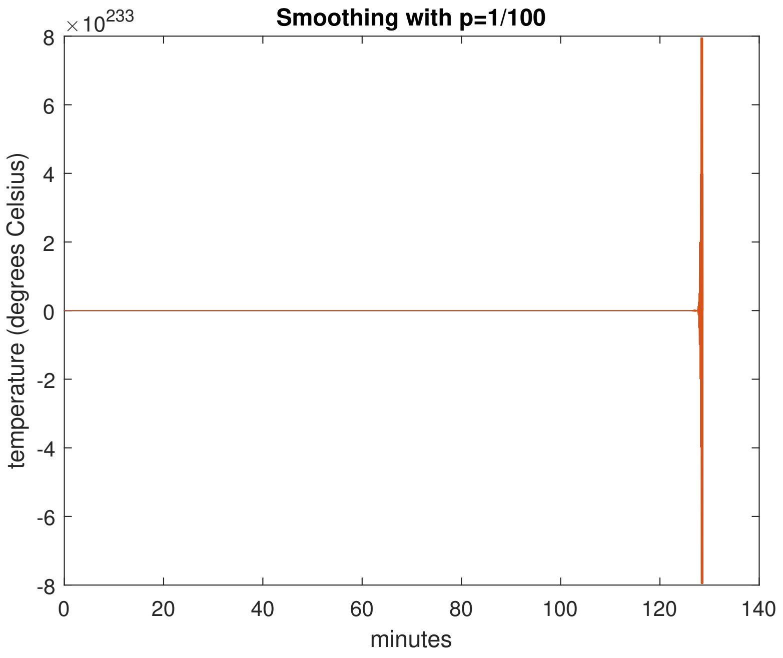


The reason why we limited the dataset to the first 1000 points is actually because, even for $p = 1/100$, which smoothes a bit too much, as shown on a previous graph, the divergence stems quickly from the increasing x.

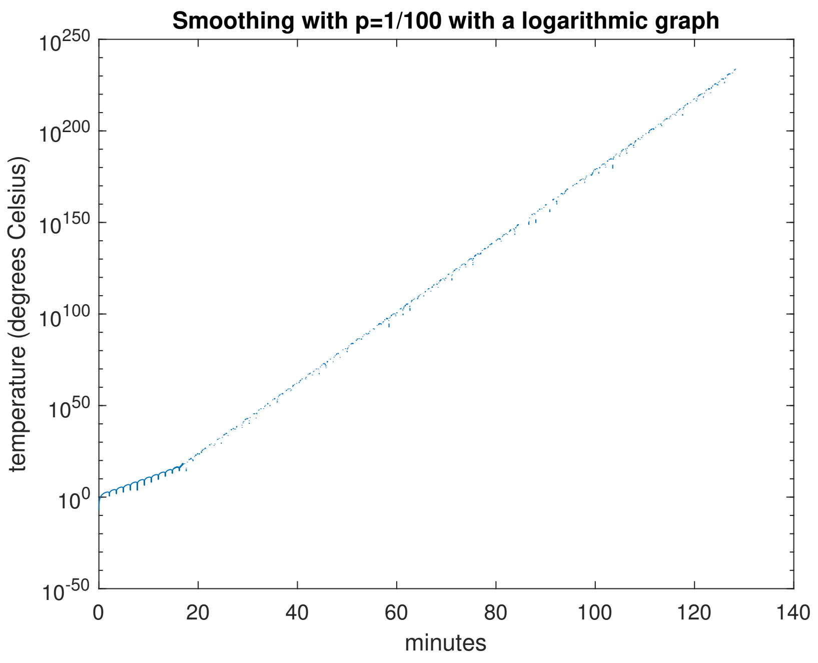
The entire dataset looks like this:



And the algorithm with $p = 1/100$ gives:



Or, alternatively, with a logarithmic scale:



Which clearly shows the exponential growth induced by the method.
So, although the algorithm might work nicely on such large intervals with symbolically computed integrals, here, with numerical data, we need to find another method so as to get a result that doesn't diverge.

4.3 Boundary Value Problem Solver

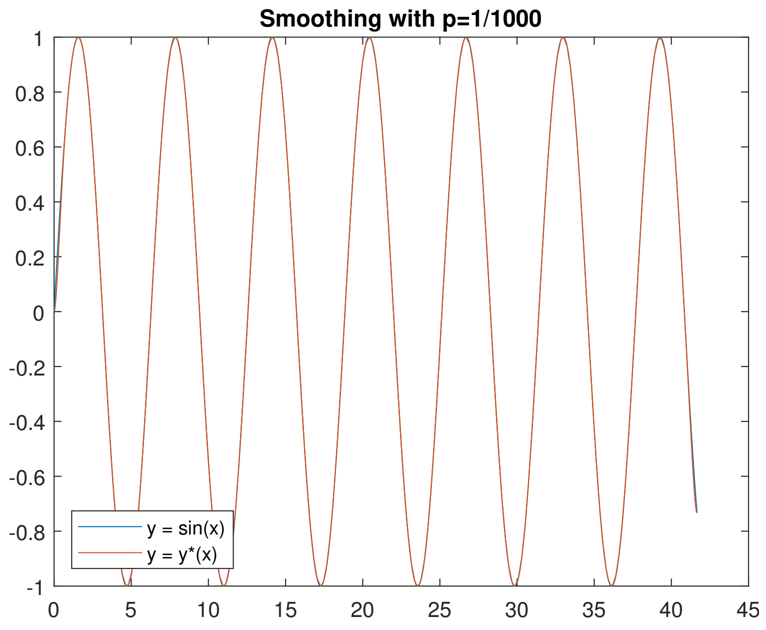
MATLAB gives us a method to solve boundary value problems directly, giving it boundary conditions and interpolating from both sides so as to stop the solution from diverging, detailed in [11].
This algorithm solves first order systems of ODEs

$$y' = f(x, y, p), \qquad a \leq x \leq b, \tag{4.7}$$

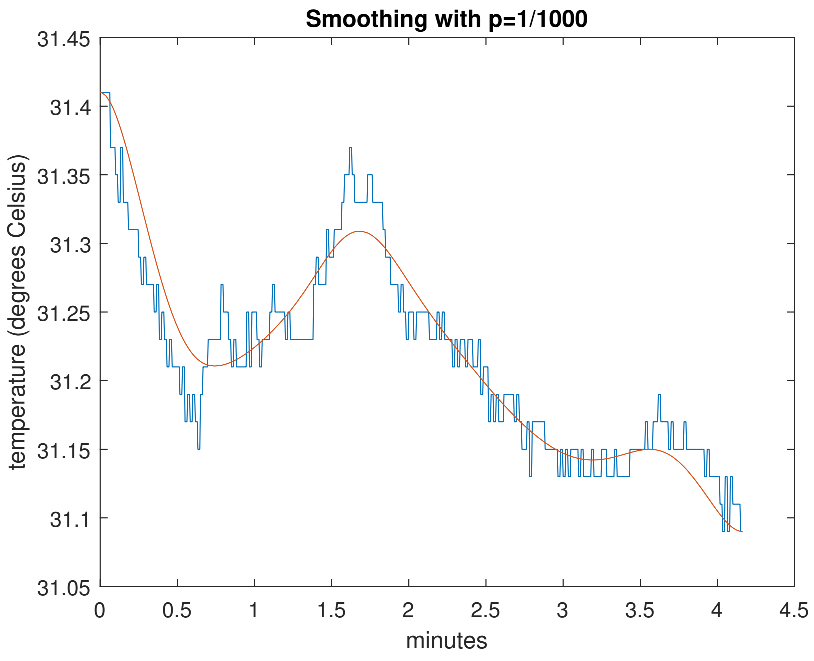
subject to two-point boundary conditions

$$g(y(a), y(b), p) = 0. \tag{4.8}$$

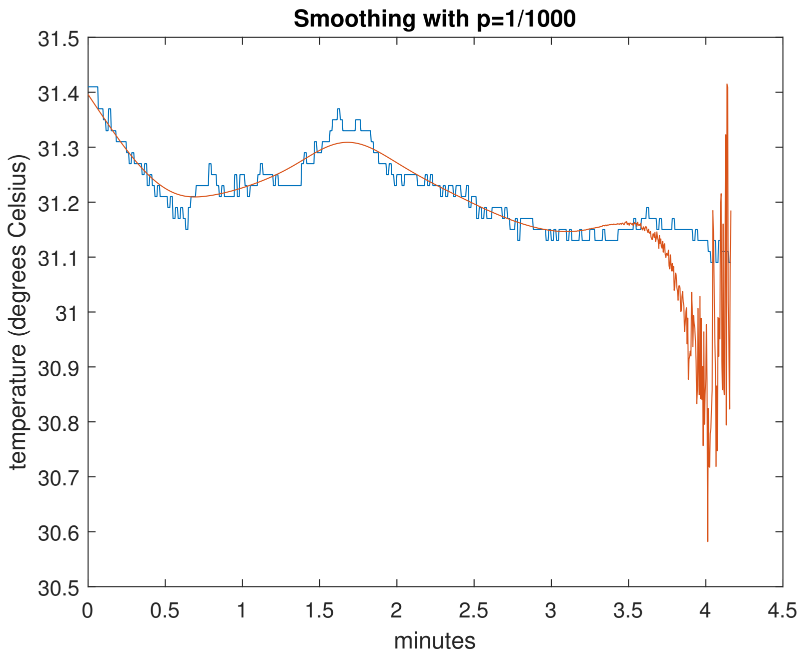
Which depends on a vector p of unknown parameters.
We assume $f(x, y, p)$ and $g(u, v, p)$ are as smooth as necessary, with in particular f being continuous and satisfying a Lipschitz condition in y. The method implemented is equivalent to the Simpson method, adjoined with a way to control for residuals, which is what should stop the result from diverging as much as before.
Applying this method gives us good results, which we can first show for a smooth function such as $\sin(x)$:



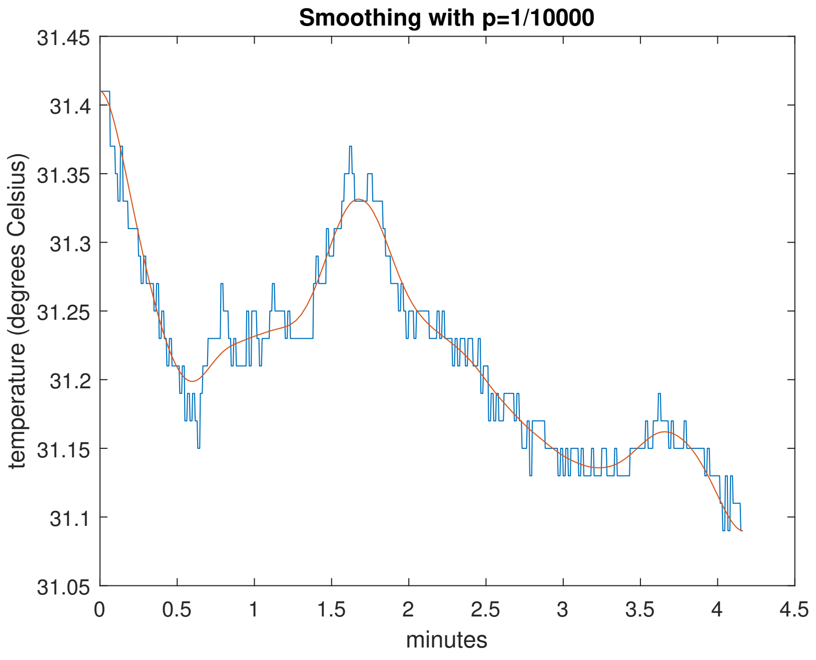
We can now apply this method to our data, with the same 1000 first data points as before:

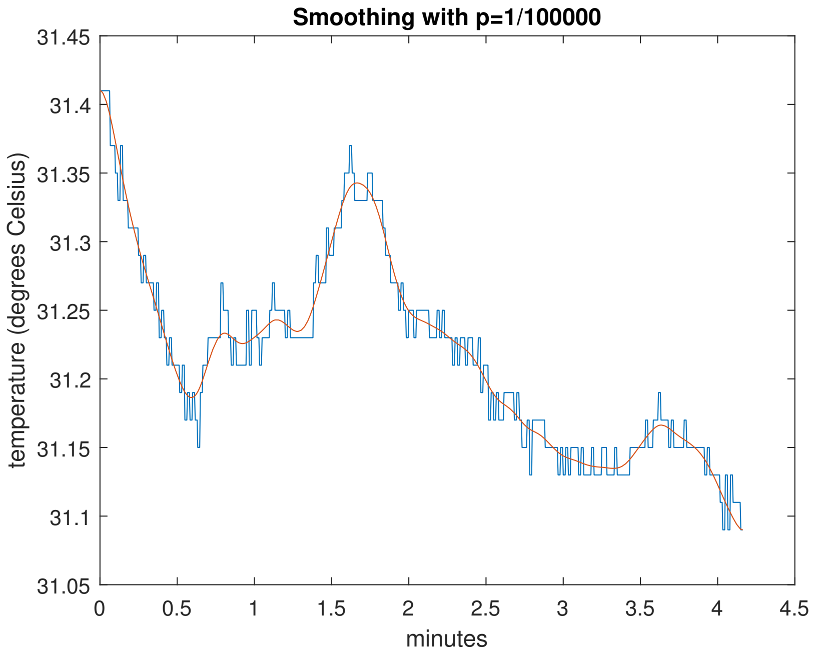


which is quite clearly way more stable than the former method, which gave us this result:

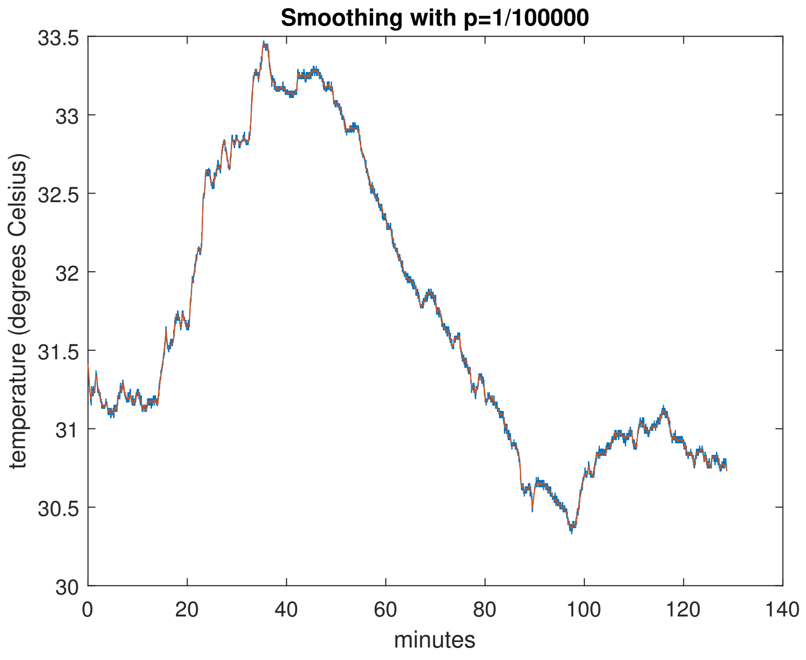


We can now increase the smoothing by decreasing p as wanted:





We can also easily apply the algorithm to the entire interval on the dataset:



Chapter 5

Discussion and Future Work

We managed to find that there's a way to use continuous data in smoothing methods, and that it works as expected. The algorithm implementation is flawed, however, due to a tendency to diverge. The next step to take would be to apply the method the solver `bvp4c` integrated to MATLAB uses to stop it from diverging as much, as the boundary conditions are enforced at several points throughout the interval - other solvers in MATLAB did not fare better than our implementation - and thus we would be able to apply the implementation to several intervals as intended.
Another thing to investigate would be, if one could find data that is already given as a function and not numerical data that has to be interpreted as such, how the method behaves solving the smoothing problem symbolically - but that is not something that would happen with the current knowledge we have.

Bibliography

[1] Magnus Egerstedt and Clyde Martin. *Control Theoretic Splines.* Princeton University Press, 2010.

[2] William E. Boyce and Richard C. DiPrima. *Elementary Differential Equations and Boundary Value Problems.* John Wiley & Sons, Inc., 2001.

[3] Jürgen Jost and Xianqing Li-Jost. *Calculus of Variations.* Cambridge University Press, 1998.

[4] Magnus R. Hestenes. *Calculus of Variations and Optimal Control Theory* John Wiley & Sons, Inc., 1966.

[5] Maurizio Brunetti. *Old and New Proofs of Cramer's Rule* HIKARI Ltd, 2014.

[6] Bernard Dacorogna. *Introduction to the Calculus of Variations* Imperial College Press, 2004.

[7] Kendall E. Atkinson. *An Introduction to Numerical Analysis* John Wiley & Sons, Inc., 1989.

[8] R. Courant and D. Hilbert. *Methods of Mathematical Physics* John Wiley & Sons, Inc., 1989.

[9] Markus Grasmair. *Basic Properties of Convex Functions* Department of Mathematics, Norwegian University of Science and Technology.
`https://wiki.math.ntnu.no/_media/tma4180/2016v/note2.pdf`

[10] Philip Schmidt, Attila Reiss, Robert Duerichen, Claus Marberger and Kristof Van Laerhoven.
Introducing WESAD, a multimodal dataset for Wearable Stress and Affect Detection
ICMI '18 Proceedings of the 20th ACM International Conference on Multimodal Interaction, 2018.

[11] Jacek Kierzenka & Lawrence F. Shampine.
A BVP Solver Based on Residual Control and the MATLAB PSE
ACM Transactions on Mathematical Software, Vol. 27, No. 3, September 2001.

Master's Theses in Mathematical Sciences 2020:E85
ISSN 1404-6342

LUNFMA-3122-2020

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/