BACHELOR

Knot positioning in splines

Minten, Jeffrey

Award date:
2022

Eindhoven university of Technology

Bachelor Final Project
2WH40

Knot positioning in splines

Author:

Jeffrey Minten
0901699
j.l.p.minten@student.tue.nl

Co-supervisors:

Paulo Serra

Rui Castro
r.m.pires.da.silva.castro@tue.nl

August 25, 2022

Abstract

Estimating data using a polynomial function can sometimes result in a polynomial with a very large degree, which are difficult to find in general. To avoid this, splines can be used. This report gives an introduction to what splines are and what they are used for. Splines are constructed using a so-called knot vector, which determines the intervals that the spline uses to estimate the data. Construction of the knot vector can prove to be a difficult and extensive task. There are already some methods known in literature to construct such a knot vector. These different methods are explained, implemented and compared to each other based on their MSE, RSS, computation time and the graphical representation of the spline that is computed using the knot vector that is output by the method. This comparison shows that some methods perform much better than others, but the overall best method is a genetic algorithm [10]. It is also found that the method that leads to the lowest RSS is also the method that performs best with regard to the MSE. Therefore optimizing the RSS for a given data set leads to the best method to be used with regard to the MSE as well. At last, this report gives the advice to investigate the optimal amount of knots as well, as this was shown by the results in this report to be possible to find.

Contents

1 Introduction

1.1 Problem Description

Estimation of a function based on collected data can be done by the use of splines. Splines are piecewise polynomials, which consist of (different) polynomials at different intervals. These intervals are divided by knots. At these knots, certain requirements have to be met. The knots are extremely important to define the approximating power of the spline. If there are fewer knots, the intervals get bigger and therefore the approximation is potentially worse. This means that intuitively more knots will give a better approximation of a function. Using more knots also means that more computational power and time are needed to estimate the spline. Also, in most cases, more intervals give a higher variance of the estimator, since the value of the function at each interval is determined while using less data per interval.

In some cases it can be beneficial to have the knots placed at certain positions such that a very smooth part of a function is approximated using less knots than a very rough part of the function. This implies that the placement of the knots is important for the approximating power of the spline.

This report will look at several methods to determine the placement of the knots. These methods will be evaluated with regard to the approximating power and the computation time of the method. This evaluation will be done for different amounts of knots. The report will eventually give an overview of the benefits of different methods with regard to each other.

1.2 Report Structure

In this report I will first explain which problem is tackled. Afterwards, some basic statistical definitions are explained to give a good start-up for the remainder of the report. Also the research questions are formulated and it is explained why these questions are relevant to answer.

Chapter 2 contains a literature overview about several methods that are constructed in other papers. These methods are explained, such that the general idea becomes clear to the reader. The technical details are not extensively explained, as these can be found in the papers that are referenced. The definitions from the introduction are used to make the explanation of each method a bit more general, such that the methods are easily comparable.

Thereafter, chapter 3 holds the implementation of the different methods. In this chapter, it is explained what starting variables are used when performing the different methods and how the results from the different methods are generated.

Then, in chapter 4 and 5 the results are shown. This is divided between a simulation part and a practical use part, where the first focuses on data samples from a known original function, while the latter focuses on data sets for which the original function is unknown. The performance of the methods are compared to each other for different amounts of knots, which are dependent on the data set. The comparison is done with regard to the RSS of all possible methods in combination with the graphical view of the results. The mean squared error is also investigated if it can be computed. Also the computation time will be investigated in the results. At last, in the simulation part the WS-MSE is also computed and compared, and it is compared to the RSS as well. This latter is done to find whether the RSS could be a good substitute for the WS-MSE to compare different methods, because the RSS can always be computed, while the WS-MSE cannot.

Afterwards, chapter 6 gives a general and summarizing conclusion based on the results from chapters 4 and 5. At last, chapter 7 discusses the potential shortcomings and/or advantages of this specific way of research. Additionally, we highlight possible paths in which future work can be performed in this chapter.

1.3 Technical explanation

1.3.1 Regression

A regression model can be used to explain a relation between two variables x and Y that are assumed to be related. We use a lower case letter to represent the covariate and a capital letter for the response to highlight that there are different assumptions made about the two: the covariate is assumed to be deterministic (not-random), while the response value is assumed to be a random quantity. A regression model is based on observations for these two variables, which are given in pairs $(x_1, Y_1), ..., (x_n, Y_n)$. For example, x_i could be the time invested in the preparations for an exam of student i and Y_i is the grade student i received for that exam. The observations of the regression model are given by the underneath formula.

$$\begin{aligned} Y_i &= r(x_i) + \epsilon_i,\ i = 1, ..., n, \\ \mathbb{E}[\epsilon_i] &= 0, \\ \mathbb{V}(\epsilon_i) &= \sigma^2. \end{aligned} \tag{1}$$

In this formula, $r : \mathbb{R} \to \mathbb{R}$ is the regression function and ϵ_i is the error at observation i. These errors are jointly independent.
The x_i's are fixed numbers and not randomly chosen. In our example this was the invested time for an exam. We can calculate the expected value of Y_i:

$$\begin{aligned} \mathbb{E}[Y_i] &= \mathbb{E}[r(x_i) + \epsilon_i] \\ &= \mathbb{E}[r(x_i)] + \mathbb{E}[\epsilon_i] \\ &= r(x_i) + \mathbb{E}[\epsilon_i] \\ &= r(x_i). \end{aligned} \tag{2}$$

From the outcome of the calculation of the expected value of Y_i, it is concluded that the Y_i are expected to be $r(x_i)$, which is the function that is interesting to estimate. Thus the goal is to estimate the function r from given $\{(x_i, Y_i)\}_{i=1}^n$.

1.3.2 Estimating a vector of parameters instead of a function

As mentioned before, $r : \mathbb{R} \to \mathbb{R}$ is the regression function. This function is unknown. Furthermore, it is unknown how complex the function r may be. It is also not known whether the family it belongs to is parametric or nonparametric.

Definition 1.1. *[1] A family of functions is called **parametric** if the behavior of members in the family is governed by a finite and fixed number of real parameters. The shape of the family is constrained by parameters which can determine variations in the shape. If not, then we speak of a **nonparametric** family.*

In this report we focus on the following families of functions:

$$\left\{ r : \mathbb{R} \to \mathbb{R} | r(x) = \sum_{j=1}^{p} \beta_j s_j(x), \beta_j \in \mathbb{R} \right\} :=< s_1, \ldots, s_p > . \tag{3}$$

All of the s_j's in the sum are chosen to be linearly independent functions.

Definition 1.2. *[2] Let V be a vector space of functions and let $R \subset V$. A set functions $\{s_j | j = 1, \ldots, p\} \in R$ is **linear independent**, if*

$$\sum_{j=1}^{p} \beta_j s_j(x) = 0 \quad \forall_{x \in \mathbb{R}} \Leftrightarrow \beta_1 = \ldots = \beta_p = 0. \tag{4}$$

The space R is defined by:

$$< s_1, ..., s_p >:= R. \tag{5}$$

Examples of sets of linear independent functions could be for instance polynomials, wavelets, trigonometric functions or splines. The latter will be used later in this report.

If p is large and $r \in V$, it is generally possible to find a set of basis functions that can effectively approximate many different regression functions. A linear combination of these independent basis functions $s_1, \ldots, s_p \in R$ gives this approximation. This approximation has a consequence, which is that instead of estimating the function r, now a vector $\boldsymbol{\beta} := (\beta_1, ..., \beta_p)$ can be estimated. Ultimately, the goal is to find an estimator for the function r which is similar to the original function. As we only have the data sample $\{(x_i, Y_i)\}_{i=1}^n$, we desire to obtain this estimator using the data sample. In the estimation of a vector it becomes possible to use the data sample, which is very desirable.

Plugging in *(3)* in the regression model gives the equation

$$Y_i = \sum_{j=1}^{p} \beta_j s_j(x_i) + \epsilon_i,\ i = 1, ..., n. \tag{6}$$

Rewriting the observations using matrices and vectors gives

$$Y = X\boldsymbol{\beta} + \epsilon, \tag{7}$$

with Y a column vector of length n, X a $n \times p$-matrix with entries $X_{ij} = s_j(x_i)$, $\boldsymbol{\beta} \in \mathbb{R}^p$ a column vector of length p and ϵ a column vector of length n. X is called the **design matrix**.

Using the rewritten representation of the data implies that finding an estimator $\hat{r} \in R$ is equivalent to finding an estimator $\hat{\boldsymbol{\beta}}$ of the parameters. The hat is used to show that this is the estimator that is dependent on the data $\{(x_i, Y_i)\}_{i=1}^n$. This rewritten model will be used most of the time in this report.

1.3.3 Quality of estimation

Given data, the regression function is approximated by an estimator $\hat{r}$. The question that raises is how good this estimator is and how this can be measured.
A number that indicates how close an estimator $\hat{r}$ is to the actual function r, is the mean squared error (MSE). Depending on the context one might consider two different definitions of MSE. The first definition focuses exclusively on the design points. This is called the within-sample MSE.

Definition 1.3. *The **within-sample mean squared error (**WS-MSE**)** of an estimator $\hat{r}$ of the function r is given by:*

$$\text{WS-MSE}(\hat{r}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left[(\hat{r}(x_i) - r(x_i))^2\right]. \tag{8}$$

Alternatively, one is often also interested in the quality of estimation outside of the design points. To investigate this quality one can take an integral over the domain of interest. The corresponding MSE is called the Mean Integrated Squared Error (MISE).

Definition 1.4. *Let $\mathbb{X}$ denote the domain of interest. The **Mean Integrated Squared Error (**MISE**)** of an estimator $\hat{r}$ of the function r is given by:*

$$\text{MISE}(\hat{r}) = \mathbb{E}\left[\int_{\mathbb{X}} (\hat{r}(x) - r(x))^2 dx\right]. \tag{9}$$

It is not possible to compute either the WS-MSE nor the MISE from data alone, as they both require knowledge of the regression function r. This means that they can both not be used directly in an inference procedure.
The MSE can be conveniently decomposed in two terms, which are the bias and variance.

Definition 1.5. *Let $x \in \mathbb{X}$ and $\hat{r}(x)$ be an estimator of the unknown $r(x)$. Then the **bias** of the estimator $\hat{r}$ at point x is defined as*

$$b(\hat{r}(x)) := \mathbb{E}[\hat{r}(x)] - r(x). \tag{10}$$

Definition 1.6. *Let $x \in \mathbb{X}$ and $\hat{r}(x)$ be an estimator of the unknown $r(x)$. Then the **variance** of the estimator $\hat{r}$ at point x is defined as*

$$\mathbb{V}(\hat{r}(x)) = \mathbb{E}\left[(\hat{r}(x) - \mathbb{E}[\hat{r}(x)])^2\right]. \tag{11}$$

Using the just defined bias and variance it can be shown that

$$\mathbb{E}\left[(\hat{r}(x) - r(x))^2\right] = b(\hat{r}(x))^2 + \mathbb{V}(\hat{r}(x)). \tag{12}$$

Using the decomposition above, we can rewrite the WS-MSE and MISE as:

$$\text{WS-MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(b(\hat{r}(x_i))^2 + \mathbb{V}(\hat{r}(x_i))\right)$$
$$\text{MISE} = \int_{\mathbb{X}}\left(b(\hat{r}(x))^2 + \mathbb{V}(\hat{r}(x))\right)dx.$$

This decomposition is useful, as it makes the computation of the MSE slightly easier. It also shows that there is a relation between the bias and the variance of an estimator with regard to the quality of estimation. The goal is to find the optimal balance between a low bias and a low variance.
In the remainder of this report the focus will be on the WS-MSE, as it is easier to evaluate using the data samples. Therefore from now on, everytime the MSE is mentioned, it means the WS-MSE.
Ideally, we want the MSE to be as small as possible. Computing the MSE requires knowledge of the unknown regression function. Therefore, it is sensible to proceed by constructing a sensible surrogate for the MSE. This surrogate can then be used as a proxy for the MSE in the inference procedure. The surrogate that is chosen is the average residual sum of squares of the estimator $\hat{r}$.

Definition 1.7. *Let $x \in \mathbb{X}$ and $\hat{r}(x)$ be an estimator of the unknown $r(x)$. Let Y be the observations that are sampled according to Equation* (1)*. Then the average residual sum of squares (RSS) of the estimator $\hat{r}$ at point x is defined as*

$$\text{RSS}(\hat{r}) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{r}(x_i)\right)^2. \tag{13}$$

Note that although the RSS is a function of $\hat{r}$, it holds that it can happen that $\text{MSE}(\hat{r}_1) \neq \text{MSE}(\hat{r}_2)$ if $\text{RSS}(\hat{r}_1) = \text{RSS}(\hat{r}_2)$. Although the RSS of the functions are equal, this does not mean that the functions are the same. This is due to the fact that they are evaluated at certain points, so the functions can vary from each other between these points. This also means that an estimating function might have a very small RSS, while it is not an accurate estimation of the original function. The RSS is no longer dependent on the unknown function r, so this can be computed from the given data.

1.3.4 Least squares estimator

We have stated that we measure the quality of fit of an estimator using the MSE. But as we are not able to compute the MSE if the original function is unknown, we instead will aim for a low RSS. Therefore, the estimator is chosen to be the function that minimizes the RSS:

$$\hat{r} = \underset{r \in R}{\arg\min}\,\text{RSS}(r). \tag{14}$$

Note that the space R was earlier defined to be:

$$< s_1, ..., s_p >:= R. \tag{15}$$

As this space is defined by linear independent functions, every r is uniquely identified by a vector $\boldsymbol{\beta} \in \mathbb{R}^p$ of parameters. This means that we can rewrite the RSS of the estimator $\hat{r}$ as:

$$\begin{aligned} \text{RSS}(\hat{r}) &\equiv \text{RSS}(\hat{\boldsymbol{\beta}}) \\ &= ||Y - X\hat{\boldsymbol{\beta}}||^2 \\ &= (Y - X\hat{\boldsymbol{\beta}})^T (Y - X\hat{\boldsymbol{\beta}}). \end{aligned} \tag{16}$$

With the formulation of the RSS($\boldsymbol{\beta}$) defined, the solution of the optimization problem can be found. This can be done by computing the gradient:

$$\nabla (Y - X\hat{\boldsymbol{\beta}})^T (Y - X\hat{\boldsymbol{\beta}}) = -2(Y - X\hat{\boldsymbol{\beta}})^T X.$$

It is then important to look at the Hessian matrix and check whether equalling the first gradient to zero yields a minimum or a maximum.

$$\nabla^2 (Y - X\hat{\boldsymbol{\beta}})^T (Y - X\hat{\boldsymbol{\beta}}) = 2X^T X.$$

The Hessian matrix is always positive semi-definite, so equalling the first gradient to zero yields a minimum. If X is a full rank matrix, there is a unique minimum. Thus it is desirable to solve this equation to find the best estimator $\hat{\boldsymbol{\beta}}$:

$$\begin{aligned} -2(Y - X\hat{\boldsymbol{\beta}})^T X = 0 &\Leftrightarrow Y^T X - \hat{\boldsymbol{\beta}}^T X^T X = 0 \\ &\Leftrightarrow Y^T X = \hat{\boldsymbol{\beta}}^T X^T X. \end{aligned}$$

After transposing both sides one finds the so-called normal equations:

$$X^T X \hat{\boldsymbol{\beta}} = X^T Y. \tag{17}$$

This means that there exists a unique solution for $\hat{\boldsymbol{\beta}}$ if $X^T X$ is invertible. If X has full rank, the solution equals $\hat{\boldsymbol{\beta}} = \left(X^T X\right)^{-1} X^T Y$.

1.3.5 Properties least squares estimator

The least squares estimator $\hat{\boldsymbol{\beta}}$ as described in section 1.3.4 has some nice properties. In this entire section we assume that X is full rank, to ensure that the least-squares estimator is uniquely defined.
The first property is that the least-squares estimator $\hat{\beta}$ yields that we find that $\hat{r}(x)$ is an unbiased estimator for $r(x)$ if the true regression function r can actually be written as

$$r(x) = \sum_{j=1}^{p} \beta_j^* s_j(x) \quad \forall_{x \in \mathbb{R}} \tag{18}$$

for some vector $\boldsymbol{\beta}^* \in \mathbb{R}^p$. First note that the estimator $\hat{r}(x)$ can be written as

$$\hat{r}(x) = \sum_{j=1}^{p} \hat{\beta}_j s_j(x) \quad \forall_{x \in \mathbb{R}} \tag{19}$$

and it also holds that

$$\hat{\boldsymbol{\beta}} = \left(\hat{\beta}_1, \ldots, \hat{\beta}_p\right)^T = \left(X^T X\right)^{-1} X^T Y. \tag{20}$$

It is now possible to show that the estimator is unbiased if $\hat{\boldsymbol{\beta}}$ is the least squares estimator.

$$\begin{aligned}\mathbb{E}[\hat{r}(x)] &= \mathbb{E}\left[\sum_{j=1}^{p} \hat{\beta}_j s_j(x)\right] \\ &= \sum_{j=1}^{p} \mathbb{E}[\hat{\beta}_j] s_j(x).\end{aligned}$$

Note that:

$$\begin{aligned}\mathbb{E}[\hat{\boldsymbol{\beta}}] &= \mathbb{E}\left[\left(X^T X\right)^{-1} X^T Y\right] \\ &= \left(X^T X\right)^{-1} X^T \mathbb{E}[Y] \\ &= \left(X^T X\right)^{-1} X^T \tilde{r}, \text{ with } \tilde{r} = (r(x_1), \ldots, r(x_n))^T \\ &= (X^T X)^{-1} X^T \begin{pmatrix} r(x_1) \\ \vdots \\ r(x_n) \end{pmatrix} \\ &\overset{\text{(if Equation (18) holds)}}{=} \left(X^T X\right)^{-1} X^T \begin{pmatrix} \sum_{j=1}^{p} \beta_j^* s_j(x_1) \\ \vdots \\ \sum_{j=1}^{p} \beta_j^* s_j(x_n) \end{pmatrix} \\ &= \left(X^T X\right)^{-1} X^T \begin{pmatrix} \sum_{j=1}^{p} \beta_j^* X_{1j} \\ \vdots \\ \sum_{j=1}^{p} \beta_j^* X_{nj} \end{pmatrix} \\ &= \left(X^T X\right)^{-1} X^T X \boldsymbol{\beta}^*, \text{ with } \boldsymbol{\beta}^* = (\beta_1^*, \ldots, \beta_n^*)^T \\ &= \boldsymbol{\beta}^*\end{aligned}$$

From this note it follows that:

$$\begin{aligned}\mathbb{E}[\hat{r}(x)] &= \sum_{j=1}^{p} \mathbb{E}[\beta_j] s_j(x) \\ &= \sum_{j=1}^{p} \beta_j^* s_j(x) \\ &= r(x).\end{aligned}$$

We have now shown that the least squares estimator results in an unbiased estimating function if the original function can be written in the form of Equation (18).

Using the least squares estimator $\hat{\boldsymbol{\beta}}$, it is also possible to find a derivation for the variance of $\hat{r}(x)$.

$$\begin{aligned}\mathbb{V}(\hat{r}(x)) &= \mathbb{V}\left[\sum_{j=1}^{p} \hat{\beta}_j s_j(x)\right] \\ &= \mathbb{V}\left[\boldsymbol{s}^T(x)\hat{\boldsymbol{\beta}}\right], \text{ with } \boldsymbol{s}(x) = \begin{pmatrix} s_1(x) \\ \vdots \\ s_p(x) \end{pmatrix} \\ &= \mathbb{V}\left[\boldsymbol{s}^T(x)\left(X^T X\right)^{-1} X^T Y\right].\end{aligned}$$

We define:

$$\boldsymbol{l}^T(x) := \boldsymbol{s}^T(x)\left(X^TX\right)^{-1}X^T. \tag{21}$$

Using this definition, we find that:

$$\begin{aligned}
\mathbb{V}(\hat{r}(x)) &= \mathbb{V}\left[\boldsymbol{s}^T(x)\left(X^TX\right)^{-1}X^TY\right] \\
&= \mathbb{V}\left[\boldsymbol{l}^T(x)Y\right] \\
&= \mathbb{V}\left[\sum_{j=1}^{n} l_j^T(x)Y_j\right] \\
&= \sum_{j=1}^{n}\mathbb{V}\left[l_j^T(x)Y_j\right] \\
&= \sum_{j=1}^{n} l_j^2(x)\mathbb{V}[Y_j] \\
&= \sum_{j=1}^{n} l_j^2(x)\sigma^2 \\
&= \sigma^2||\boldsymbol{l}(x)||_2^2.
\end{aligned}$$

Finding a derivation for the bias and variance of $\hat{r}(x)$ enables us to find a derivation for the mean squared error, which becomes:

$$\begin{aligned}
\text{WS-MSE}(\hat{r}) &= \frac{1}{n}\sum_{i=1}^{n}\left(b(\hat{r}(x_i))^2 + \text{Var}(\hat{r}(x_i))\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\left((0)^2 + \sigma^2||\boldsymbol{l}(x_i)||_2^2\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\sigma^2\boldsymbol{l}^T(x_i)\boldsymbol{l}(x_i) \\
&= \frac{\sigma^2}{n}\sum_{i=1}^{n}\boldsymbol{s}^T(x_i)\left(X^TX\right)^{-1}X^T\left(\boldsymbol{s}^T(x_i)\left(X^TX\right)^{-1}X^T\right)^T \\
&= \frac{\sigma^2}{n}\sum_{i=1}^{n}\boldsymbol{s}^T(x_i)\left(X^TX\right)^{-1}X^TX\left(X^TX\right)^{-1}\boldsymbol{s}(x_i) \\
&= \frac{\sigma^2}{n}\sum_{i=1}^{n}\boldsymbol{s}^T(x_i)\left(X^TX\right)^{-1}\boldsymbol{s}(x_i) \\
&= \frac{\sigma^2}{n}\sum_{i=1}^{n}\begin{pmatrix}s_1(x_i) & \dots & s_p(x_i)\end{pmatrix}\left(X^TX\right)^{-1}\begin{pmatrix}s_1(x_i)\\ \vdots \\ s_p(x_i)\end{pmatrix} \\
&= \frac{\sigma^2}{n}\sum_{i=1}^{n}\begin{pmatrix}X_{i1} & \dots & X_{ip}\end{pmatrix}\left(X^TX\right)^{-1}\begin{pmatrix}X_{i1}\\ \vdots \\ X_{ip}\end{pmatrix} \\
&= \frac{\sigma^2}{n}||L||_2^2, \text{ with } L = X\left(X^TX\right)^{-1}X^T.
\end{aligned}$$

The derivation above assumes that the basis functions are fixed, which means they are not dependent on the data. The methods that are described in chapter 2, however, are trying to optimize the knot placement. This causes the basis functions to be variable, as these depend on

the placement of the knots. Therefore this derivation might not be useful for solving the knot position problem.

1.3.6 Splines

In this report the main focus will be on the functions s_i being splines. To better understand splines, first the concept of piecewise polynomials should be explained. Suppose that an interval is divided into k subintervals and a polynomial is assigned to each subinterval. These polynomials together form a piecewise polynomial.

Definition 1.8. *(Piecewise polynomials)[4]*
Let $a = v_0 < v_1 < \ldots < v_k = b$ and $\Delta = \{v_i\}_{i=0}^{k}$. Δ partitions interval $[a, b]$ into k subintervals, $I_i = [v_i, v_{i+1})$, $i = 0, 1, ..., k-1$.
*$\mathcal{P}_m$ is the space of polynomials of order m and is given by $\mathcal{P}_m = \{f : \mathbb{R} \to \mathbb{R} | f(x) = \sum_{i=1}^{m} c_i x^{i-1}$, $c_1, ..., c_m\}$. Given a positive m, let $\mathcal{PP}_m(\Delta) = \{$f: there exist polynomials $f_0, f_1, ..., f_{k-1}$ in $\mathcal{P}_m$ with $f(x) = f_i(x)$ for $x \in I_i, i = 0, 1, ..., k-1\}$ where $\mathcal{PP}_m(\Delta)$ is called the space of **piecewise polynomials** of order m on the partition Δ.*

Piecewise polynomials are not necessarily continuous. This is why splines are used. Splines are build up from piecewise polynomials, but they are continuous everywhere. Also a spline of order m should be $m-1$ times continuously differentiable.

Definition 1.9. *(Polynomial splines)[4]*
$\mathcal{S}_m(\Delta) = \mathcal{PP}_m(\Delta) \cap C^{m-2}[a, b]$. $\mathcal{S}_m(\Delta)$ is called the space of polynomial splines of order m on the partition Δ. Also $C^{m-2}[a, b] = \{$f: the $(m-2)^{th}$ derivative of f belongs to $C[a, b]\}$, where $C[a, b] = \{$f: f is continuous at each x in $[a, b]\}$.

In the above definitions we have certain partitions that define the subintervals. The simplest way to define these interval partitions is by indicating the endpoints of the intervals. These points are denoted as *knots*. This means that knots uniquely represent an interval partition.

1.3.7 B-splines

As the space R that is desired is a space of splines, an appropriate basis for R is needed. The specific splines that will be used for the functions $s_i(x)$ are B-splines. To define the B-splines basis functions it is first necessary to extend the interval partition as defined below.

Definition 1.10. *(Extended partition) [4] Let $a < v_1 < v_2 < \ldots < v_k < b$, $\Delta = \{v_i\}_{i=1}^{k}$ and m be given. Suppose $z_1 \leq z_2 \leq \ldots \leq z_{2m+k}$ is such that $z_1 \leq \ldots \leq z_m \leq a, b \leq z_{m+k+1} \leq \ldots \leq z_{2m+k}$ and $z_{m+1} < \ldots < z_{m+K} = v_1, v_2, \ldots, v_k$, with each v_i represented once.*
Then we call $\tilde{\Delta} = \{z_i\}_1^{2m+k}$ an extended partition associated with $\mathcal{S}_m(\Delta)$.

In this report it is used that $z_1 = \ldots = z_m = a$ and $z_{m+k+1} = \ldots = z_{2m+k} = b$. This extended partition results in a larger number of knots. Some of these knots share the same value and these knots are called multiple knots. For the purposes of this report it is assumed that there are no multiple knots, besides the knots that are at the boundary points.This means that all interior knots that are computed have multiplicity 1.
It is now possible to look at how the B-splines are constructed. This is done recursively. First the initial values for $i = 1, \ldots, k + 2m - 1$ are defined:

$$B_i^1(x) = \begin{cases} 1 & \text{if } z_i \leq x < z_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

Then all other B-splines are determined recursively by:

$$B_i^{m^*}(x) = \frac{x - z_i}{z_{i+m^*-1} - z_i} B_i^{m^*-1}(x) + \frac{z_{i+m^*} - x}{z_{i+m^*} - z_{i+1}} B_{i+1}^{m^*-1}(x), \tag{23}$$

for $m^* \leq m$ and $i = 1, \ldots, k + 2m - m^*$. Note that it is possible that the denominator in this recursion becomes zero if you are at a multiple knot. If this is the case, it is defined that the fraction becomes zero if the denominator would be equal to zero.

In this report the focus is on the cubic splines, which means that the basis functions for the B-splines are the functions

$$\{B_i^4(x), i = 1, \ldots, k + 4\}.$$

1.3.8 Penalty

The problem that is addressed in this project focuses on the amount of knots and the placement of these knots, as well as on the accuracy of the estimator. The RSS is known to be a good estimator for the MSE if the dimension is small. If p becomes very large, the RSS behaves quite similar to the bias instead of the MSE in general. Therefore a smoothness can be added such that the variance that contributes to the MSE is also taken into account sufficiently. This means that the estimator should be smooth enough to avoid a large penalty, but it should also be close enough to the real value to avoid a large RSS. By adding this penalty, we no longer want to minimise just the RSS. We aim to minimise the addition of the RSS and the penalty to obtain the desired estimator.

The penalty will be a roughness penalty. This roughness penalty makes sure that the result from the minimization does not become a function that simply interpolates the data points. This penalty is in this report given the following form:

$$\lambda \int_{\mathbb{X}} (\hat{r}''(x))^2 dx. \tag{24}$$

This leads to the original problem being to find $\hat{r}$ such that:

$$\hat{r} = \underset{r \in C^2[a,b]}{\arg\min} \left\{ \frac{1}{n} \sum_{i=1}^{n} (Y_i - r(x_i))^2 + \lambda \int_{\mathbb{X}} (r''(x))^2 dx \right\}. \tag{25}$$

The solution to the problem above is known to be a natural spline with knots at the data points. However, we are interested in the solution if there are less knots available then there are data points. This problem now becomes finding $\hat{r}$ such that:

$$\hat{r} = \underset{r \in R}{\arg\min} \left\{ \frac{1}{n} \sum_{i=1}^{n} (Y_i - r(x_i))^2 + \lambda \int_{\mathbb{X}} (r''(x))^2 dx \right\}. \tag{26}$$

As we are now only investigating splines with a certain amount of knots, we can rewrite the penalty as:

$$\int_{\mathbb{X}} \left(\frac{d^2}{dx^2} (\boldsymbol{s}(x)^T \hat{\boldsymbol{\beta}}) \right)^2 dx, \tag{27}$$

with

$$\boldsymbol{s}(x) = \begin{pmatrix} s_1(x) \\ \vdots \\ s_p(x) \end{pmatrix}. \tag{28}$$

The penalty can then also be given the form

$$\lambda \hat{\boldsymbol{\beta}}^T \Omega \hat{\boldsymbol{\beta}}. \tag{29}$$

In this form the $\hat{\boldsymbol{\beta}}$ is the estimator of $\boldsymbol{\beta}$, λ is the parameter that assigns how much the penalty contributes to the error and Ω is the matrix that defines the error with

$$\Omega_{ij} = \int_{\mathbb{X}} s_i''(x) s_j''(x) dx. \tag{30}$$

The calculation of this Ω is omitted here, as it is done using basic mathematical operations. This report focuses on cubic B-splines, thus we can rewrite the penalty as:

$$\lambda\hat{\boldsymbol{\beta}}^T\Omega\hat{\boldsymbol{\beta}}, \text{ with } \Omega_{ij} = \int_{\mathbb{X}} (B_i^4)''(x)(B_j^4)''(x)dx. \tag{31}$$

It is now at last possible to rewrite the original problem of finding $\hat{r}$ as a problem of finding $\hat{\boldsymbol{\beta}}$ such that:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{(k+4)}}{\arg\min} \left\{(Y - X\boldsymbol{\beta})^T(Y - X\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\Omega\boldsymbol{\beta}\right\} \tag{32}$$

It is then found that the value of $\hat{\boldsymbol{\beta}}$ equals

$$\tilde{\boldsymbol{\beta}} = \left(X^TX + \lambda\Omega\right)^{-1} X^TY. \tag{33}$$

In the penalized regression model, the λ determines the amount of smoothing. This λ can be chosen by minimizing the generalized cross-validation score. In this case the effective degrees of freedom is given by $\nu = tr(X(X^TX + \lambda\Omega)^{-1}X^T)$. Here the tr represents the trace of the matrix, which is the sum of the diagonal elements. This results in

$$GCV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{\mathbf{r}}(x_i)}{1 - \frac{\nu}{n}}\right)^2. \tag{34}$$

Although now there is a expression for λ, still the problem regarding the amount and placement of the knots is open.

1.4 Research questions

In statistics, linear regression is often used to estimate data with a function, such that the relation between two or more variables can be captured. In some cases, variables have a relation that is far from linear. This ensures that the use of standard linear-regression techniques is not possible. Therefore non-parametric methods are more adequate to use, but many of them cannot capture different regimes within a data set adequately. An example of such a data set is given below:

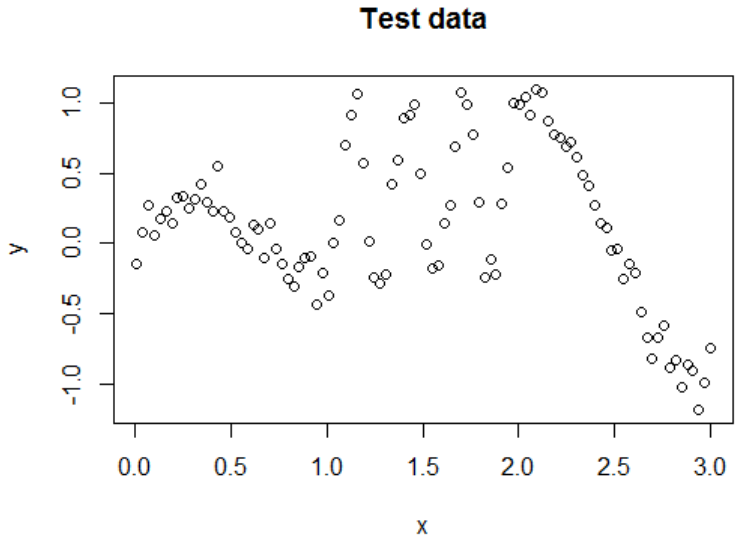


Figure 1: A data set that shows a relation between two variables that seems to be non-linear.

In the example in Figure 1 the relation between the variables seems to be different in different intervals. This means that the standard non-parametric methods are not adequate enough. This is shown in Figure 2 by the splines with 13 regularly placed knots, and natural splines with knots at the data points.

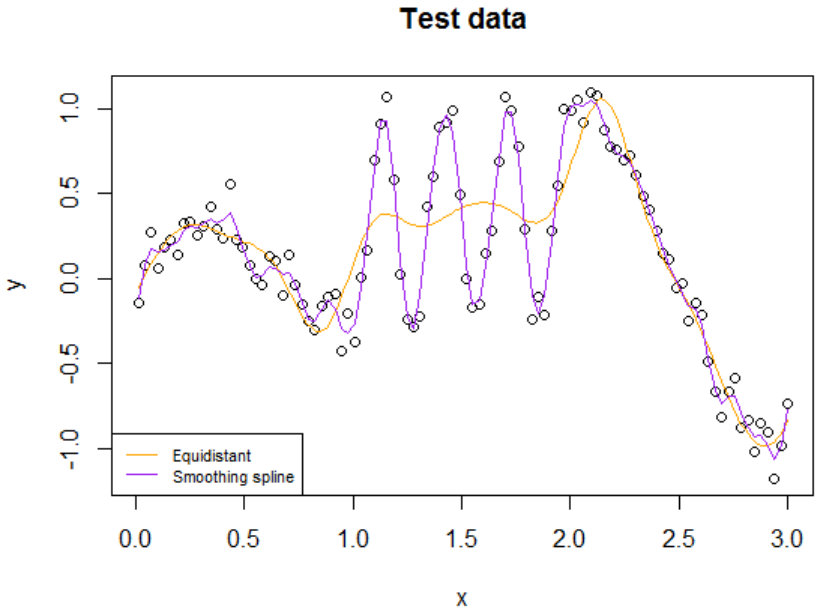


Figure 2: Approximating function of test data using natural spline and a spline using knots at 13 equidistant locations.

These methods yield functions that are uniformly regular, which is not beneficiary in this example. The smoothing spline in Figure 2 seems to be performing pretty well, but it can be seen that especially in the interval $[0, 1]$ the data seems to be interpolated by this method. Therefore, it is interesting to investigate whether this resembles the original function and if this can be improved if this is not the case.
It is also possible to clairvoyantly place the same amount of knots as the equidistant method uses. These knots are placed at locations $(0, 0.5, 0.99, 1, 1.01, 1.14, 1.28, 1.42, 1.56, 1.7, 1.85, 1.99, 2, 2.01, 3)$. Note that this are 15 locations instead of 13, but as the boundaries are obliged to be knots, we have 13 interior knots, which is the same as used by the equidistant method here. The locations of the clairvoyant knots are carefully chosen. It can be seen that at $x = 1$ and $x = 2$ there are multiple knots closely to each other. This is done to ensure that the spline becomes variable with regard to the derivative in these parts of the data interval. These positions are chosen because they seem to be the parts where the relation between the variables changes. Furthermore, for each remaining interval there are knots chosen equidistant, with more knots if the data seems to be fluctuating more. An example for this data set is given in Figure 3.

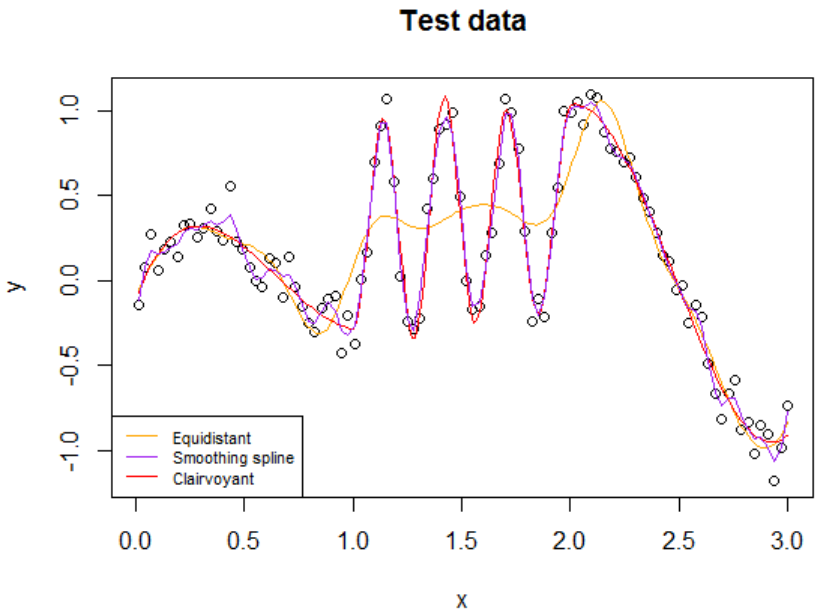


Figure 3: Approximating function of data set by a spline using knots at clairvoyantly chosen locations.

As can be seen in Figure 3, one is able to get a different representation by having clairvoyantly placed knots. As this data set is constructed from a known function, it is also possible and interesting to look at the resemblances between the constructed splines and the original function. This is done in Figure 4.

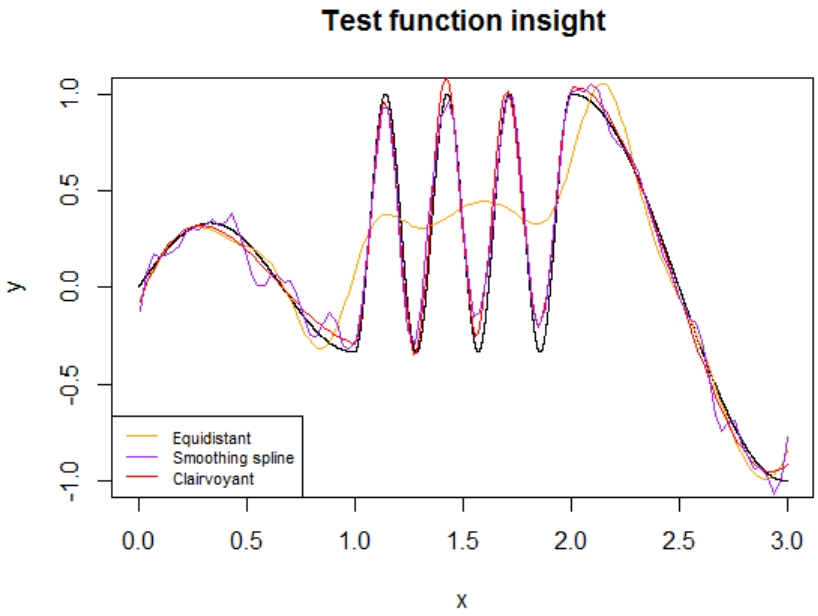


Figure 4: Approximating function of data set by a spline using knots compared to the original function that the data set was sampled from.

In Figure 4 it can be seen that the spline using the clairvoyantly chosen knots shows more resemblance to the original function than the other two methods. For the equidistant method, it can be seen that this method especially has a rather terrible estimation in the interval $[0.9, 2.1]$, as it does not capture the fluctuation of the original function in this interval. The smoothing spline method, on the other hand, performs quite badly in the intervals $[0, 1]$ and $[2, 3]$, as it fails to capture the smoothness of the original function in these intervals. These errors in both methods are especially due to the fact that they are more focused on estimating the data points rather than the original function itself. As this latter is the most interesting part of the estimation, it is important to investigate different methods to place the knots.
This report will investigate data-driven knot placement methods and compare their performances objectively using the MSE, as well as the computational complexity. As the MSE requires knowledge of the original regression function, it is also investigated whether the RSS is a good substitute for the MSE. This leads to the following research questions:

- Given a certain amount of knots, what is the ideal placement of them?
- What algorithm for selecting the locations of the knots is most effective/the best?
- How do the algorithms perform compared to choosing equidistant knots and the concept of natural splines?
- Is the RSS of the estimated spline with regard to the data sample a good substitute for the MSE of the estimated spline with regard to the original function?

2 Literature overview

In literature the problem of the placement of knots has already been extensively examined. Below a literary overview is given.

2.1 Optimization methods

First of all, it is possible to use optimization to find the optimal placement of the knots. There are two methods described below that try to simultaneously optimize the placement of the knots and the fitted parameters of the estimating spline that is used. Both the described methods already specify the estimating spline to become the least squares estimator of the data, such that the problem only becomes dependent on the placement of the knots.
As described before, the B-spline basis is dependent on the placement of the knots. This means that the design matrix X is also different if the knots are placed differently. Therefore we now use $X(z)$ to indicate that we use the design matrix based on knot vector z. This is done by defining the RSS in a new way:

$$RSS(\boldsymbol{\beta}, \boldsymbol{z}) = ||Y - X(\boldsymbol{z})\boldsymbol{\beta}||_2^2 \tag{35}$$

Using this, the desired objective is to solve:

$$(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{z}}) = \underset{\boldsymbol{\beta}\in\mathbb{R}^{(k+4)}, \boldsymbol{z}\in\mathbb{R}^{(k+8)}}{\arg\min} ||Y - X(\boldsymbol{z})\boldsymbol{\beta}||_2^2. \tag{36}$$

For a given set of knots we know how to optimize the choice for $\boldsymbol{\beta}$, so ultimately the optimal placement of knots is given by:

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}\in\mathbb{R}^{k+8}}{\arg\min} ||Y - X(\boldsymbol{z}) \left((X^T(\boldsymbol{z})X(\boldsymbol{z}))^{-1}X^T(\boldsymbol{z})Y\right)||_2^2 := \underset{\boldsymbol{z}\in\mathbb{R}^{k+8}}{\arg\min}\, q(z). \tag{37}$$

Optimizing this RSS now becomes optimizing the function $q(\boldsymbol{z})$, which is only dependent on $\boldsymbol{z}$. $\boldsymbol{z}$ itself is constructed based on $\boldsymbol{v}$ with four additional knots at each boundary of $\mathbb{X}$. $\boldsymbol{v}$ is the vector of k knots that is computed by the methods below. Afterwards the additional 8 knots are added to find the extended knot vector $\boldsymbol{z}$. This means that it is only necessary to find an optimal knot vector $\boldsymbol{v}$, as the other knots in $\boldsymbol{z}$ are fixed.
The methods below both use nonlinear optimization techniques. The methods differ at this point, as they use different optimization techniques.

2.1.1 Gauss-Newton with Marquardt modification [5]

The technique that is proposed by Jupp [5] is the Gauss-Newton method with Marquardt's modification (GNM). This requires the Jacobian matrix,

$$J(z) = \left[-\frac{\delta X(\boldsymbol{z})}{\delta z_j}\left(X(\boldsymbol{z})^T X(\boldsymbol{z})\right)^{-1} X(\boldsymbol{z})^T Y + X(\boldsymbol{z})\frac{\left(X(\boldsymbol{z})^T X(\boldsymbol{z})\right)^{-1} X(\boldsymbol{z})^T}{\delta z_j} Y\right]_{j=1,\ldots,k+8}. \tag{38}$$

The way in which this computation is done is explained by Golub and Pereyra [6].
This technique is easy to implement and very well-known within the optimization environment.

For this problem a few transformations are required to be compatible for the GNM technique. These transformations are also made to have a higher probability that local minima that are not the global minimum are avoided. These transformations are based on the intervals between knots, instead of their exact location.

Using this transformation and some additional matrices based on the intervals between knots, the rows of J are transformed one at a time. Using the latest matrix J the knot vector can be extracted. A more thorough explanation on this method can be found in the paper [5].

2.1.2 Cutting angle [7]

In the paper by Beliakov [7], the author focuses on a deterministic global optimization technique to solve the nonlinear part of the knot position problem. The method used to solve this problem is the method of cutting angle. It is assumed that there are no knots outside $\mathbb{X}$ to enable the use of this technique. The cutting angle method has the idea of constructing a saw-tooth cover of the objective function. This saw-tooth cover is a max-min type function, which is constructed such that it always underestimates the estimated function. The objective of the method is to find the saw-tooth cover with the highest global minimum.

More technical explanation on this method can be found in the paper [7].

This cutting angle method is subject to a large computational effort, which makes it less favorable to use. However, it does guarantee that it reaches the global optimum, rather than getting stuck in a local optimum, as it checks all viable options for the support vectors. This method is according to the paper [7] more favourable if the amount of knots is reasonably small.

2.2 Knot vector adaptation methods

Besides methods which rely on optimization-based techniques to solve the problem, there are also some methods that first compute a set of knots in a rough way, after which this knot set is used to find the optimal knot positions in a more thorough way. Two of these techniques are described here.

2.2.1 Bisecting method to find the coarse knots [8]

The first technique that is explained in this section is the method proposed by Dung [8]. This method starts from a few knots and increases the amount of knots for the first steps, which are explained below:
Start by splitting $\mathbb{X}$ in half. Fit a third degree polynomial to each new interval and evaluate the RSS on this interval. If this RSS is sufficiently low, this interval is considered appropriate. If this is not the case, the interval will be split in half again. This process is repeated until all intervals are considered appropriate or if the maximum amount of available knots is reached.
The second step checks all pairs of consecutive pieces based on if they can appropriately be considered to belong to a single piece. If this is the case, the pieces are merged and the knot in the middle is erased.
Afterwards the knots are shifted with the intention to expand the large pieces and possibly reduce the small pieces to a single location, such that an additional knot can be removed.
The knot vector that is given as output is checked for each piece again whether the interval is considered appropriate. If all intervals are considered appropriate, this knot vector is determined to be the optimal knot vector that this method gives. If not, the algorithm is repeated until the maximum number of available knots is used by an intermediate knot vector or if all pieces are considered appropriate. This method is easily understood as it mostly consists of splitting the intervals in half. As it evaluates every piece of interest in each step, it might get a quite large computation time. This is a summarized explanation of the technique. For the full technical details the paper [8] can be read.

2.2.2 Two-Step method [9]

Another technique that uses a modification of an initial knot vector is described by Kang [9]. This method will be described in more detail, as it gives an interesting insight in how the method performs and what similarities it has with earlier described methods. The method has the strategy to solve the problem in two steps, which are:

1. Compute a set of active knots from a dense initial knot vector.

2. Adjust the positions of the active knots. If two of the active knots are at the same position, one of them is called redundant and is removed. The final knot vector is the vector obtained after the adjustment of the positions. This knot vector is then used to compute the spline.

These two steps will be explained more thoroughly.
The first step of this method is about the computation of a set of active knots. These are computed using an initial knot vector $U = \{z_0, ..., z_{n+4}\}$, which has knots that are usually chosen uniformly in $\mathbb{X}$. Recall that the function $\boldsymbol{r}(x)$ satisfies Equation (3), with the functions $s_j(x)$ being B-splines. After that, define $\boldsymbol{r}^{(\gamma)}(x)$ as the γ^{th} order derivative of the function $\boldsymbol{r}(x)$. We also define the jump of the 3^{rd} order derivative of $\boldsymbol{r}(x)$ at knot z_j as

$$|\boldsymbol{r}^{(3)}(z_j^+) - \boldsymbol{r}^{(3)}(z_j^-)|, \text{ where } \boldsymbol{r}^{(3)}(z_j^+) = \lim_{x \uparrow z_j} \boldsymbol{r}^{(3)}(x) \text{ and } \boldsymbol{r}^{(3)}(z_j^-) = \lim_{x \downarrow z_j} \boldsymbol{r}^{(3)}(x).$$

Now a sparse optimization model with initial knot vector U is solved. This optimization model is given by

$$\begin{aligned} &\min_{\beta_i \in \mathbb{R},\ i=0,\dots,n} ||J\mathbf{r}^{(3)}(x)||_1, \\ &\text{such that } \sum_{i=1}^{n} (\mathbf{r}(x_i) - Y_i)^2 \leq n \cdot \theta, \\ &\text{where } ||J\mathbf{r}^{(3)}(x)||_1 = \sum_{j=4}^{n} \left| \mathbf{r}^{(3)}(z_j^+) - \mathbf{r}^{(3)}(z_j^-) \right| \end{aligned} \tag{39}$$

Note that in the optimization model θ is chosen by the user and corresponds to a fixed tolerance to control the quality of fit. The knots that have a non-zero jump at the 3^{rd} derivative are called the active knots. This means that the 3^{rd} derivative is discontinuous at the active knots. The active knots are now contained in a new vector $\tilde{U} = \{t_1, \dots, t_p\}$.

After the active knots are found, the positions of the active knots are adjusted. This is again done in two steps.
First of all, all intervals in the active knot vector are investigated whether the interval is applicable to be merged into a single knot. This is done as follows:
For each interval $[t_i, t_{i+1}]$ $(i = 1, \dots, p-1)$, the point $t^* = \frac{t_i + t_{i+1}}{2}$ is inserted to get a new knot vector $U^* = \{t_1, \dots, t_i, t^*, t_{i+1}, \dots, t_p\}$. Using this new knot vector, the optimization problem is solved again. Then it is checked for which of the three knots t_i, t^* or t_{i+1} the jump of the m^{th} derivative is the minimal one. If this minimum is not at t^*, then the interval is considered applicable and will be narrowed down to a single knot.
The second step is the narrowing down into a single knot for each applicable interval. This is done for one interval at a time. For each applicable interval, either knot t_i or t_{i+1} is replaced by t^*. This replacement is done based on the optimal replacement with regard to the least square estimating spline. This process is repeated for this interval until the interval is small enough. When this is the case, the two endpoints of the interval are merged into a single knot, which is placed in the middle of these endpoints. The new knot vector U^* is updated accordingly. After all applicable intervals are merged into a single knot, the latest knot vector U^* is considered the resulting knot vector.

2.3 Random search methods [10]

At last, two methods that are based around random searches are considered. Both these methods are described in the same paper, written by Spiriti [10]. In this paper, the knots are chosen such that the generalized cross-validation criterion (GCV) is minimized. For our use, only the optimization of the GCV with regard to the knot vector is dealt with, as the optimization with regard to the smoothing parameter is not necessary when the knot vector is computed.

2.3.1 Blind search with golden section adjustment

The first method that is considered in the paper [10] is a blind search with a golden section adjustment. The blind search repetitively picks k knot locations at random in $\mathbb{X}$. The golden section adjustment (GSA) is a deterministic search algorithm that looks for a nearby local minimum of the objective function. The adjustment is chosen to be used on a knot-by-knot basis. This means that the RSS is treated as a function of a single knot location with the other $k-1$ knot locations fixed. The possible interval $\tilde{\mathbb{X}}$ that this function is defined on are the neighbouring knots. It is assumed that the function does not have any extrema besides a minimum within $\tilde{\mathbb{X}}$.
The GSA is a one-dimensional optimization algorithm that computes the optimal knot location in a rough way.
The optimization scheme is carried out in four passes through the knots. First forwards, then twice backwards and at last forwards again. There is also a version that works for penalized regression that uses the GCV as a replacement for the RSS. Full details on this method can be found in the paper [10].

2.3.2 Genetic algorithm

A 'population' is constructed using elements of the solution space. Then the algorithm has three parts, which are selection, crossover and mutation. The selection optimizes the function, the crossover combines two elements of the 'population' to yield another element and mutation randomly modifies elements of the 'population'. In the problem of knot position the selection allows only the best knot sets to survive. The crossover chooses two 'parent' knot sets uniformly at random from the survivors. A child population is formed from these 'parent' knot sets which is of equal size to the parent population. The mutation chooses a random knot set uniformly from the survivors and a random knot z_l is sampled and replaced by a random number chosen uniformly in (z_{l-1}, z_{l+1}).
A detailed pseudo-code description is given in [10].

3 Implementation

In this chapter, the implementation of all different methods for the simulation and for the comparison with regard to the known data sets is explained.
For the implementation we have used several steps. This is due to the fact that some papers have used Matlab to implement their technique, while others have used R. As it is very useful to have the original code from the authors of the papers, these codes will not be tampered with.
Firstly, R is used to generate data samples if necessary. As was explained before, there are two functions that are considered to be needed to generate data samples for.
Afterwards, these data samples are stored in a way that it can be accessed for all programming languages that have original code. These various languages are used to generate the resulting knot vectors for the corresponding methods. Also the time of computation is computed simultaneously, such that this can be compared as well.
Lastly, the resulting knot vectors are used to compute splines in R, which are then used to compute both the RSS and the WS-MSE. This latter is only possible for the samples for which the original function is known.
For the more general steps R is used, as this program has nice features with regard to plotting figures and it also has several built-in features that are useful for this project. The implementation of the different methods are explained for each of the methods individually. This contains the choice of starting values as well as the way in which the implementation is written.

3.1 Gauss-Newton with Marquardt modification

The paper [5] in which this method is described has a very thorough explanation on how the technique is performed and which steps have to be taken to perform the algorithm. These steps are therefore implemented, including a built-in function in R that computes the Jacobian matrix, such that no mistakes are made in that part. The technique also has the benefit that it only requires the amount of knots as input. This means that the user of this method does not have to choose any other input variables himself, which results in a very general algorithm.
Unfortunately, during the implementation of this algorithm in R, quite some errors occurred. This mainly consisted of difficulties regarding the Jacobian matrix and extracting the new knot vectors from the updated Jacobian matrix. This causes the results from this method to be unreliable, and after extensive research the possible mistakes that were made were unfortunately not found. It was decided to exclude this method from the results due to this unreliability, because it is undesirable to draw conclusions that might be incorrect with regard to this knot-placing method.

3.2 Cutting angle

The paper [7] itself has a clear explanation on how the method works. Using this detailed explanation, an algorithm is written in R. There are only a few initial values that have to be chosen for this method, which are the amount of support vectors that are used to find a solution in the unit simplex, as well as the accuracy that is used to find the optimal solution of the minimalization problem. This latter was done by stating a maximum amount of steps this method takes, such that the accuracy of the method is good. Both these starting values are chosen by the user, as the paper itself does not have a clear explanation of these choices. It was decided that there are 25 support vectors used, and that the steps are repeated 10 times. These values where determined after checking different values. We found that this amount of support vectors gives enough variability to determine an optimal solution. The amount of times the steps are repeated have also shown to be sufficient, as after this number of steps the improvement did not remain significant enough to be worth the additional computation time.

3.3 Bisecting method

The appendix of the paper in which this technique is explained [8] contains the Matlab-code of this technique that was written by the author of this paper [8]. This code that was found in the appendix is used to implement this method.
This method unfortunately uses quite a lot of initial values that have to be chosen by the user, besides the amount of knots and the degree of the spline. These contain the maximum fitted error of a single piece B-spline (0.01), the minimal joining angle between two pieces of the spline (1) and the maximum smoothness of the spline (-1). These values are chosen in this report equal to the values that are chosen by the author of the paper [8] and are given in brackets behind the explanation of the initial values before.
Besides these values, also the number of evaluation times to calculate the discontinuity case and the maximum number of loops for the Gauss-Newton solving are required, but these are not given in the paper. These are chosen to be both equal to 50, as this ensures that the spline is not discontinuous and that the improvements of the Gauss-Newton solving remain significant.
Unfortunately, after the implementation of this method it was found that converting the results from R to Matlab and vice versa gave some problems. These problems were not possible to solve in due time. Therefore, as the results from this method are not representative in this case, it was decided to omit the results from this method from the report. This was again done to prevent drawing wrong conclusions from results that were not properly generated.

3.4 Two-step method

The paper [9] that describes the Two-step method contains very extensive pseudo-codes, which are converted to R code. This method requires a few initial values that have to be chosen by the user, which are a tolerance an a maximum amount of loops. The tolerance represents a value for which the method is considered to be significantly close to the original function to terminate the loop. If this tolerance is not reached, also a maximum amount of loops is chosen. This amount is chosen to be 50, as this reduces the computation time, while it still gives enough time to reach the tolerance. This tolerance is chosen equally for all data sets, and chosen the same as the author of the paper [9], which is 0.025.

3.5 Blind search with golden section adjustment

The author of this paper [10] has embedded the code in R in a single function himself. This means that there is no further programming involved in this method whatsoever. Also the starting values that are needed for this method are equal to the starting values that are used in the paper [10]. This concerns the amount of times a random set of knot vectors is computed and the size of each individual random set. The amount of times a random set is equal to 1000, and the size of each individual random set was chosen to be equal to 200.
We unfortunately did not manage to load the correct packages into R to be able to use the pre-programmed function, hence this method is manually programmed according to the pseudo-code in the report. Comparing the results of this programmed function with the actual function gave no significant differences, so the conclusions drawn from this method can still be interpreted and deemed relevant. These results where compared by performing the manually programmed code on the data sets that are given in the paper [10]. These results were very similar to the results presented in the paper [10].

3.6 Genetic algorithm

This technique is from the same paper [10] as the BSGS method, and for this method also the author has embedded it in a single function in R. This method has more starting values than the BSGS method. After research, it was determined that we should not choose the same starting values as in the paper [10], as this would lead to a very large computation time. Also checking

whether the method improves significantly by using the large starting values stated in the paper [10] showed that this is not the case. The starting values concern the amount of times the algorithm is ran (5/20) , the amount of generations per time (10/200), the amount of knot vectors in each generation (100/1000), the amount of survivors from each generation (10/100), the amount of mutations in each generation (100/1000) and how much the mutated knot vectors are mutated (0.02/0.02). The values that are between brackets after each starting value used state the value used in this paper, and the value used in the paper [10] respectively. Using the values from this report reduces the computation time by approximately 800 times with regard to the starting values the author from the paper [10] used.
The embedded function by the author [10] is contained in the same package as the BSGS method, which means that unfortunately the same problem occurred as before. Therefore this problem is handled similar as was done for the previous method. Also checking whether the manually programmed code was similar to the code by the author [10] was done in a similar way.

3.7 Equidistant knots

The technique of using equidistant knots is quite simplistic. Firstly there are two knots determined at the boundaries of $\mathbb{X}$. So $v_0 = \min_{x \in \mathbb{X}} x$ and $v_{k+1} = \max_{x \in \mathbb{X}} x$. Then the remaining k interior knots are chosen such that $|v_i - v_{i-1}| = \frac{v_{k+1} - v_0}{k+1} \quad \forall_{i \in 1,\ldots,k}$. This means that the positions of the knots are predefined for each amount of knots and each interval $\mathbb{X}$. This method is only dependent on the number of knots used, so no other initial values have to be inserted. This technique is programmed in R using the simplistic representation given.

3.8 Smoothing spline

The smoothing spline is a natural cubic spline with knots at the data points. This spline minimizes the penalized RSS, as stated by Theorem 5.73 in the book by Wasserman [3]. This smoothing spline is the estimation that is used in general if the amount of knots is not bounded. Therefore it is interesting to look at the comparison between the different methods and this generally used technique to be able to see how much accuracy is possibly lost by being bounded in the amount of knots that are available. The smoothing spline is a built-in function in R, which is used to compute for the different data sets. The smoothing spline is highly dependent on the penalization term used to minimize the penalized RSS. The built-in function in R determines this penalization term using leave-one-out cross validation, which is a good data-driven method to determine a good value for this term. Therefore this built-in feature is used to determine the penalization term.

3.9 Amount of knots

The amount of knots for which the methods are evaluated are chosen dependent on the data set. Not all data sets consist of an equal amount of data points, so using the same amounts of knots for evaluation is not desirable. For all data sets, the methods are at least evaluated for 10 knots. This is done as this amount of knots is manageable to compute, and it also enables us to compare between different methods whether there are significant differences when using the same amount of knots.
There are two more amounts of knots for which the methods are evaluated. These amounts are dependent on the data set. It is chosen that the first amount is 30/40% of the data set size and the second amount is 60/70% of the data set size. This leads to the amount of knots that are chosen per data set that can be seen in Table 1.

Table 1 The amount of knots for evaluation of the data sets.

Data set	Standard knots	30-40 percent	60-70 percent
Doppler	10	30	70
Test function	10	30	70
Titanium	10	20	35
Motorcycle	10	30	70

4 Simulation

Now all methods are explained, it is interesting to look how these perform with regard to each other. The comparison between the methods is done with regard to the MSE. Recall that the MSE in this report means the WS-MSE, as this is the one that is easier to compute. We will also compare using the RSS, which is as close to the MSE as we are able to compute if the original function is not known. Using this, we want to see whether the RSS gives a good representation of the performance of the method with regard to the MSE. Also the computation time of each method is investigated, as this enables us to compare the time investments to the accuracy. The comparisons are done with regard to the different amount of knots from Table 1 for different data samples. The data samples will be explained in the first part of this chapter.
After these explanations, the methods are compared to each other. This is done for two different constructed data sets that were sampled from an original function with two different noise standard deviations σ. For both these functions 200 samples are taken and for each sample the methods are performed. The results will be supported with visual representations of them.
All methods that are mentioned in chapter 2 are compared to each other as well as to two other techniques, which are choosing equidistant knots and the smoothing spline with knots at the data points.

4.1 Data samples

To be able to have a good comparison between the methods, several different data samples are used. These data samples are explained here, as well as the reasoning of choosing these data samples. Note that the data samples are constructed using:

$$\begin{aligned} Y_i &= f(x_i) + \epsilon_i,\ i = 1, ..., n, \\ \mathbb{E}[\epsilon_i] &= 0, \\ \mathbb{V}(\epsilon_i) &= \sigma^2. \end{aligned} \tag{40}$$

In this equation, the Y_i represent the sampled data points, f represents the original function that was sampled from, the x_i represent the values at which the data points are sampled and the ϵ_i is the randomly sampled noise for this data point with noise distribution $\epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0; \sigma^2) \quad \forall_{i=1,...,n}$.
The two functions that data is sampled from are described below. The first function that is considered is the so-called Doppler-function, which can be seen in Figure 5.

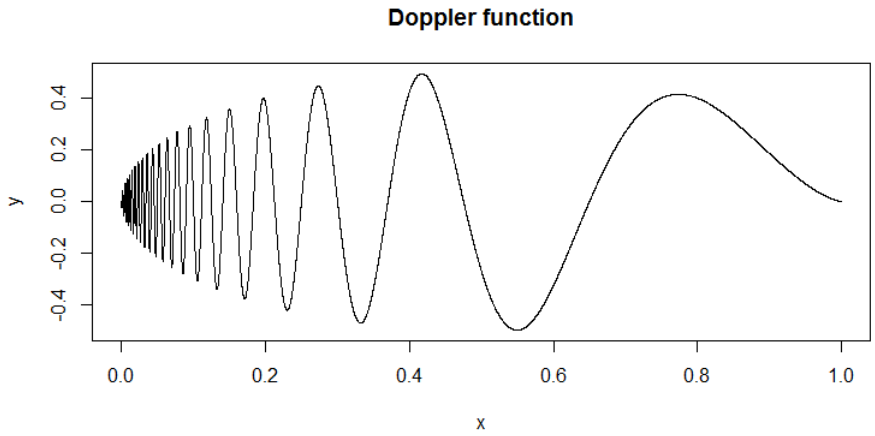


Figure 5: The Doppler function.

This function is given by the formula

$$Doppler(x) = \sqrt{x(1-x)}\sin\left(\frac{2.1\pi}{x+0.05}\right).$$

This function will be used to sample 200 observations from with noise standard deviation σ. These samples are taken for two different values of the noise standard deviation, which are $\sigma = \sqrt{0.01} = 0.1$ and $\sigma = \sqrt{0.001}$. These values are chosen such that there is a difference of a factor 10 with regard to the variance. Comparison with regard to the MSE will be done for this data sample, as this truly shows how accurately a method estimates the original function. This function is chosen as it shows different characteristics at different parts of the interval of the function. Therefore it is difficult to estimate this function using a single polynomial, thus splines are useful to estimate this splinear function.
The second function that is sampled from can be seen in Figure 6.

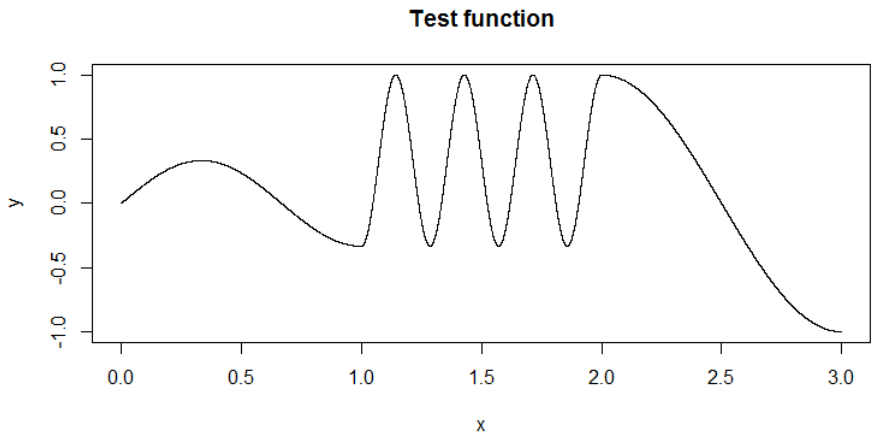


Figure 6: The Test function.

This function is given by the formula

$$Testfunc(x) = \begin{cases} \frac{1}{3}sin(\frac{3\pi}{2}x) & \text{, if } 0 \leq x < 1 \\ \frac{2}{3}cos(7\pi x) + \frac{1}{3} & \text{, if } 1 \leq x < 2 \\ cos(\pi x) & \text{, if } 2 \leq x \leq 3. \end{cases}$$

For this function the data is sampled similarly as for the Doppler function. Also the same values of noise standard distribution σ are used to sample the data. This function is interesting to consider, as it can be seen that the function has different characteristics at the different intervals. However, in contrary to the Doppler function, there are no parts of the function that are obviously rough, like the first part of the Doppler function. It is therefore interesting to see whether the methods possibly perform better at this function than at the Doppler function.

4.2 Results

In this section the two data sets that have been sampled from an original function are evaluated. This is done for each data set individually and for all different amounts of knots and methods. For each data set, the 200 samples are all used to perform the different methods. For these samples, the RSS and MSE can be computed, for which also the mean and standard deviation will be investigated. The same holds for the computation time of the different methods. For the visual results, sample 1 is used. In the plots of the results, also the eventual knot vector of the best performing method with regard to the MSE is shown on the x-axis. Which method is the best performing can also be seen by the color of the ticks that represent the knot vector on the x-axis.

4.2.1 Doppler function

First the so-called Doppler function is investigated. The results for the MSE when the data is sampled using noise standard deviation $\sigma = 0.1$ can be found in Table 2. This table shows some interesting results.

Table 2 The MSE of the different methods on data samples from the Doppler function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0264	0.0005	0.0144	0.0008	0.0103	0.0013
Cutting angle	0.0357	0.0102	0.0149	0.0054	0.0112	0.0026
Two-step method	0.0259	0.0067	0.0116	0.0023	0.0111	0.0015
BSGS	0.0347	0.0097	0.0158	0.0052	0.0174	0.0838
Genetic algorithm	0.0133	0.0014	0.0073	0.0014	0.0106	0.0014
Smoothing spline	mean			sd		
	0.0098			0.0018		

First of all, it can be seen that the MSE is significantly lower for each method if the method uses 30 knots instead of 10 knots. This causes us to think that more knots is generally better to minimize the MSE. However, when taking a look at the use of 70 knots, this shows that the improvement is far less than expected. For the genetic algorithm and the BSGS method, the performance even becomes worse when the amount of knots is increased from 30 to 70. This gives the impression that for each method there is an optimal amount of knots to be used, such that adding or removing one knot does not lead to a significant improvement.
Another observation that leads to this impression is the difference in performance between the knot-placing methods and the equidistant knots. As can be seen for 10 knots, all methods except for the genetic algorithm perform worse or similar with regard to the MSE than the equidistant knots. For 30 knots, these same methods perform similar or better with regard to the MSE than the equidistant knots. For 70 knots, it again can be seen that these methods perform worse than equidistant knots. This supports the idea that the knot-placing methods have an optimal amount of knots for which they perform best.
The last interesting thing that we will mention in this report is the performance of the genetic algorithm. It can be seen that this method has a much lower MSE than all of the other methods when using 10 or 30 knots. It even performs better than the smoothing spline method if 30 knots are used, which is the method that minimizes the penalized RSS. This causes us to believe that the genetic algorithm is favourable to use. It should, however, also be noted that the relative difference in performance between the genetic algorithm and the other methods decreases as the amount of knots increases. Therefore it is possible that the genetic algorithm is not the optimal performing knot-placing method if the amount of knots becomes quite large.

As mentioned before, the Doppler function is known, which causes us to be able to compute the MSE. However, in most cases the original function is unknown, which means that it is impossible to compute the MSE. Therefore we want to investigate whether the RSS is a good substitute for the MSE to determine which method estimates the original spline the best. We have thus computed the RSS for the Doppler function sample with noise standard deviation $\sigma = 0.1$, for which the results can be found in Table 3.

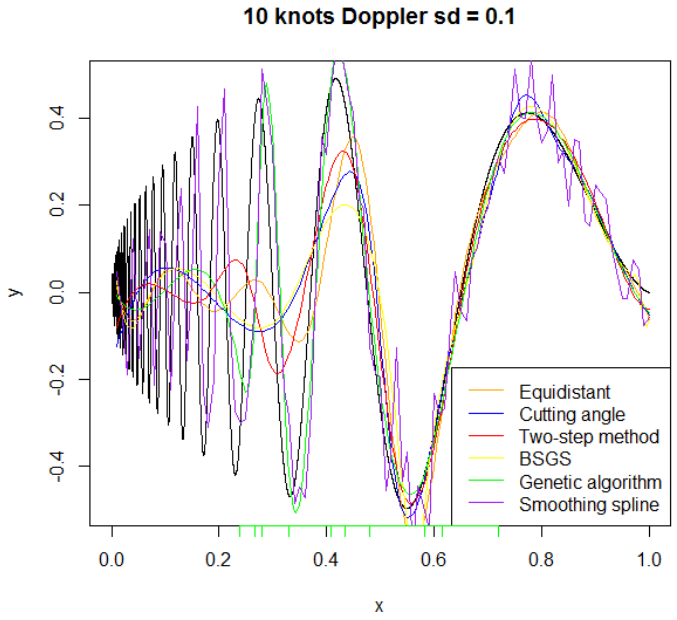
Table 3 The RSS of the different methods on data samples from the Doppler function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0320	0.0037	0.0161	0.0022	0.0030	0.0008
Cutting angle	0.0488	0.0114	0.0172	0.0056	0.0055	0.0025
Two-step method	0.0311	0.0071	0.0121	0.0028	0.0028	0.0010
BSGS	0.0404	0.0104	0.0174	0.0055	0.0154	0.1382
Genetic algorithm	0.0181	0.0025	0.0071	0.0012	0.0015	0.0005
Smoothing spline	mean			sd		
	0.0062			0.0047		

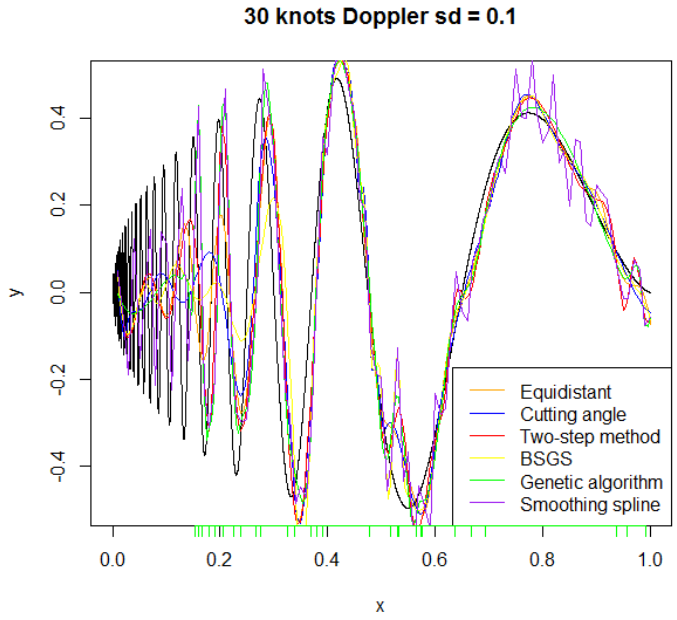
Table 3 shows some similar results to Table 2, as still the methods that have an MSE close to each other also have an RSS that is similar. Again the genetic algorithm performs better than the other methods.
There is also one difference that is quite remarkable, but also well explainable. With the MSE we saw that the difference in performance between using 30 or 70 knots was not large. For the RSS this difference is much larger, as this seams to mean that the RSS becomes significantly smaller as the amount of knots keeps growing. This can be explained by the properties of the RSS. The RSS does not take into account the smoothness of the spline, while the MSE does. Thus as the amount of knots grows, the data points will be estimated more accurately, because all methods are solely focused on minimizing the RSS. This causes that we are getting closer to interpolation of the data if more knots are used, as this is the aim of all of the methods. The estimating spline might, however, also become less smooth. This latter is not taken into account by the RSS, so this gives the impression that a spline performs better with more knots, but this is not necessarily true, as the MSE might become worse if the data is interpolated more accurately.

To test the idea mentioned above, we will also take a look at some visual results. In Figure 7 the plots of the estimating splines outputted by the different methods can be found.

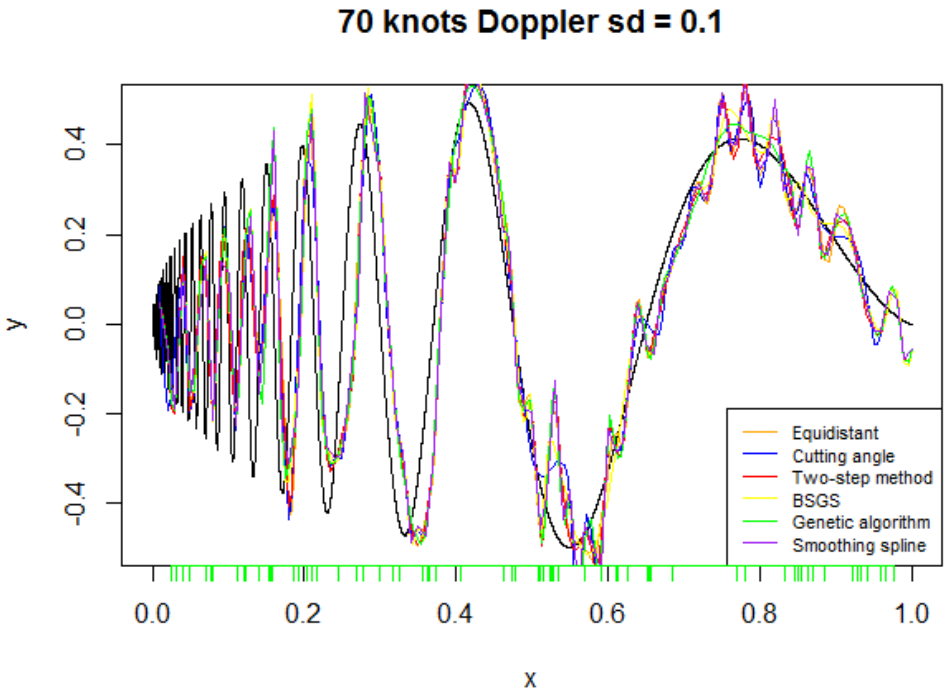
In Figure 7 the original function that is sampled from is given in black. As can be seen by the plots, having a higher number of knots does not necessarily mean that the original function is captured better. The purple line indicates the smoothing spline, which takes all data points as input. Especially for $x \geq 0.5$ it is seen that this estimating spline does not capture the trend well, as it gives a much more spiky look in this interval.
When comparing Figures 7a, 7b and 7c, also some interesting things can be concluded. For all methods that are included in this report, the accuracy of the estimating spline seems to become better for $x \leq 0.5$ as the amount of available knots grows. However, it also appears to be the case that the accuracy becomes worse for $x \geq 0.5$ as the amount of knots grows. This latter can be seen particularly well in Figure 7c. Also the methods seem to perform worse in the interval $x \in [0.5, 0.6]$ in Figure 7b than they do in the same interval in Figure 7a.
These visual results support the idea that the knot-placing methods have an optimal amount of knots for which it performs best, as there seems to be a trade-off between being accurate at the rougher parts of the function or at the smoother parts.
Taking a closer look at the knot placement also leads to some noteworthy remarks. As can be seen in Figures 7a and 7b, the best performing method starts to perform similar to the original function where the knot vector starts. This means that to accurately estimate the original function, the knot vector should also include knots in the lower parts of the interval. However, in Figure 7c, it can be seen that the large amount of knots in the higher parts of the interval shows that the data is overfitted. This leads to the conclusion that in this part of the function less knots should be used.
At last, it is not only important how well a method performs, but also the time it takes to perform the method. Therefore the methods are all also timed and these results can be found in Table 4.



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

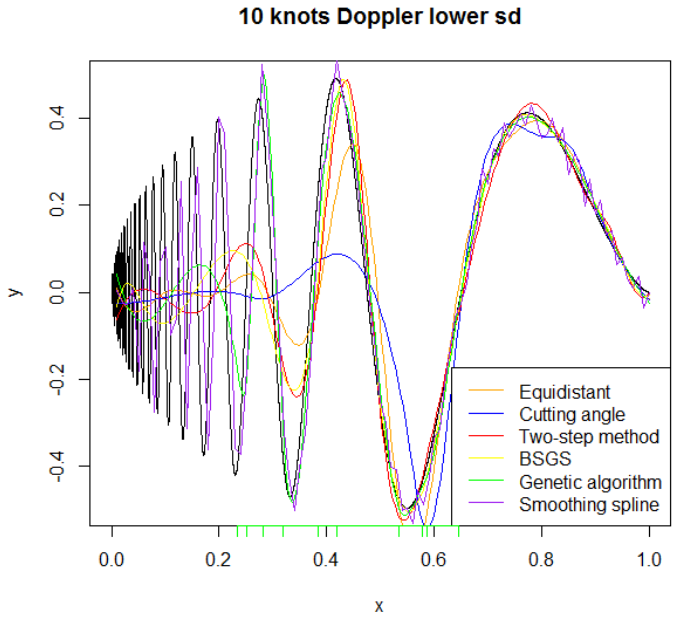
Figure 7: Comparing the methods on the Doppler function data samples ($\sigma = 0.1$).

Table 4 The computation time (in seconds) of the different methods on data samples from the Doppler function ($\sigma = 0.1$).

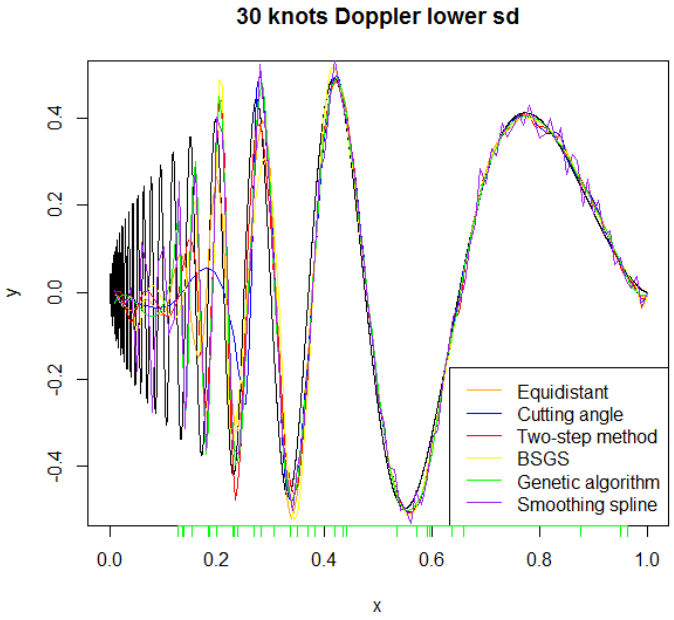
Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0008	0.0086	0.0002	0.0017	0.0004	0.0026
Cutting angle	4.853	0.3905	5.117	0.4003	6.013	0.4676
Two-step method	3.001	0.6389	17.405	1.581	91.424	5.988
BSGS	0.6438	0.0846	1.092	0.1147	2.798	0.1686
Genetic algorithm	14.574	1.768	26.862	2.735	71.203	2.452
Smoothing spline	mean			sd		
	0.0011			0.0043		

The results in Table 4 show that for all methods it holds that the computation time increases as the amount of knots increases. While this trend is quite similar for all methods, the term with what it increases is different for all methods. It can be seen that the Two-Step Method is much more influenced by the amount of knots for the computation time than for instance the Cutting angle method. This should be taken into account when choosing a method, as for larger amount of available knots some methods might take long to compute.
It can also be seen that the time it takes to compute the methods if there are 10 knots available is quite similar for all of them, besides the BSGS method and the genetic algorithm. The BSGS method performs much faster than all other methods in this report, and it also does not seem to be influenced much by the amount of available knots. Thus this method might be more suitable if a quick computation is desired. The genetic algorithm, on the other hand, is quite slow compared to the other methods, especially if fewer knots are used. Because this method performs best when using 10 or 30 knots, this method is preferable to use if the accuracy is more important, and the computation time required is less important.

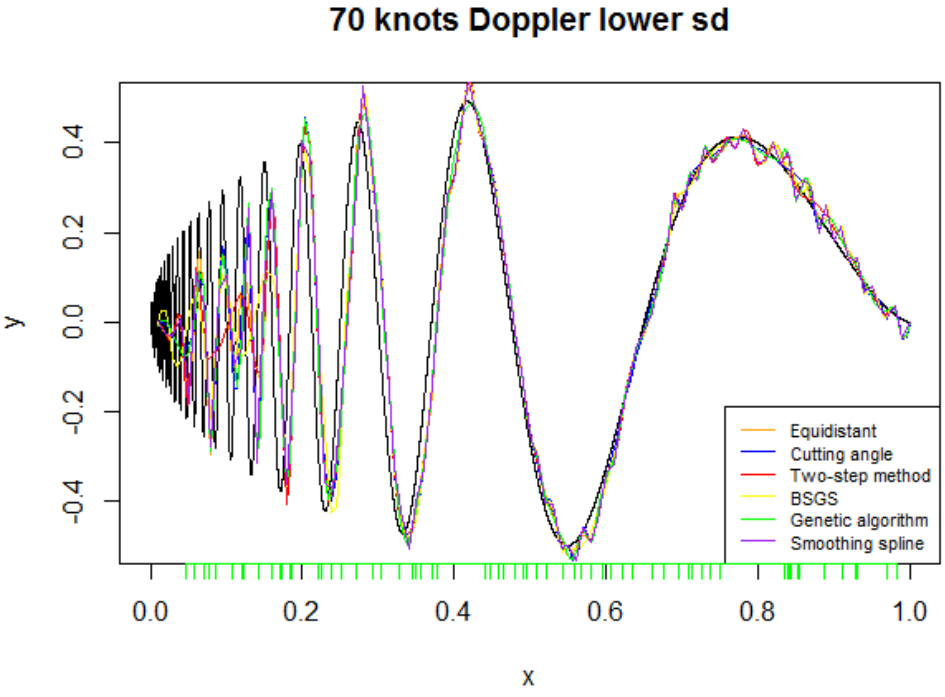
All of the aforementioned results for the Doppler function are from the samples that were computed using noise standard deviation $\sigma = 0.1$. These results are also computed for a lower noise standard deviation of $\sigma = \sqrt{0.001}$. We will investigate whether a lower noise standard deviation influences the results with regard to the MSE, RSS and computation time. The full results on the lower noise standard deviation can be found in Appendix A.2. These results are not included here for readability purposes, but the most interesting differences and similarities are explained.
For the MSE, using different noise standard deviations show a lot of similarities with regard to which method performs better than the other methods. For 10 knots there are no significant differences between the different noise standard deviations, but for 30 knots the MSE's are a little bit lower if the noise standard deviation is lower. At last, looking at 70 knots, there is a rather large difference in the MSE's, as a lower noise standard deviation results in a lower MSE. Also the impression that there might be an optimal amount of knots for which the methods perform best is less apparent if the noise standard deviation is lower.
This can be explained by the properties of the MSE. The MSE can be rewritten as the sum of the variance and the squared bias of the estimator. As the data sample contains points that are generated with a lower noise standard deviation, the variance of the estimating spline with a higher number of knots will likely not increase as much as when the noise standard deviation is higher. Therefore the bias takes a relatively larger role in the MSE and this is likely to decrease if the amount of knots increases.
Taking a further look at the differences between the different noise standard deviations, we find that the values of the RSS are smaller if the noise standard deviation of the data sample is smaller. This is especially the case if the amount of knots increases. The RSS thus becomes smaller if the noise standard deviation of the sample is smaller. Furthermore there are no significant differences if the noise standard deviation is altered.
The visual results for a noise standard deviation $\sigma = \sqrt{0.001}$ can be seen in Figure 8.
Looking at the visual results in Figure 8, some similarities can be found to those in Figure 7.



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 8: Comparing the methods on the Doppler function data samples ($\sigma = \sqrt{0.001}$).

Still the behavior of the different methods appears to be that the function is estimated better for the rougher parts if the amount of knots is larger and better for the smoother parts if the amount of knots is smaller. This behavior is less apparent in Figure 8, however, because the fluctuations at the smoother interval are not as large as in Figure 7. This is likely due to the lower noise noise standard deviation of the sampling, which causes the data points to be closer to the original function. This causes the trade-off between estimating the rougher parts or estimating the smoother parts to shift a little, as estimating the smoother parts in a rougher way looks more like the actual function than it did when the noise standard deviation was higher. Thus changing the noise standard deviation does not alter the behavior of the different methods, but it does possibly change the optimal amount of knots used to estimate the original function. Also the same conclusions with regard to the knot placement can be drawn as that were drawn from Figure 7, as the plots in Figure 8 show the same properties. The overfitting of the data in Figure 8c is less apparent, but this is due to the fact that the noise standard deviation of the samples is lower, so the data points are closer to the original function.
At last the computation time for the lower noise standard deviation is compared to the computation time for the higher noise standard deviation. Differing the noise standard deviation doesn't show any significant differences, which means that the noise standard deviation of the sample does not influence the computation times of the different methods.

4.2.2 Test function

As was done by the aforementioned Doppler function, also the Test function is investigated. It is interesting to see how the different methods perform in this case, as the properties of this function are slightly different from the properties of the Doppler function. The Test function does for instance not have a fluctuating part like the Doppler function has in the interval $[0, 0.1]$. Therefore the difference between the smoother parts and the rougher parts of this function are smaller. It is interesting to see what happens when this difference is reduced.
First we will therefore take a look at the MSE of the different methods with regard to the Test function with the same original noise standard deviation $\sigma = 0.1$. These results can be found in Table 5.

Table 5 The MSE of the different methods on data samples from the Test function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0769	0.0006	0.0050	0.0010	0.0080	0.0013
Cutting angle	0.0841	0.0119	0.0258	0.0178	0.0080	0.0027
Two-step method	0.0770	0.0040	0.0066	0.0032	0.0082	0.0014
BSGS	0.0801	0.0093	0.0242	0.0168	0.0080	0.0017
Genetic algorithm	0.0478	0.0025	0.0055	0.0012	0.0086	0.0013
Smoothing spline	mean			sd		
	0.0049			0.0011		

Comparing the results in Table 5 to the results in Table 2 shows quite some interesting differences. It can especially be seen that the difference in performance between using 10 or 30 knots is much larger in the Test function. This means that this Test function is much more difficult to capture using fewer knots than the Doppler function. This indicates that the optimal amount of knots to use is probably dependent on the original function and its characteristics. Furthermore, there are some other notable things in Table 5. It can be seen that both the Cutting angle method and the Two-step method keep improving in performance as the amount of knots increases. This is remarkable, as this difference can only be seen for using 30 knots. For using 10 or 70 knots, these methods are similar with regard to performance to the other methods. For the other methods we see that they do not improve with regard to the MSE if the amount of knots is changed from 30 to 70 knots. This is similar to the results in Table 2, so it still supports the impression that there

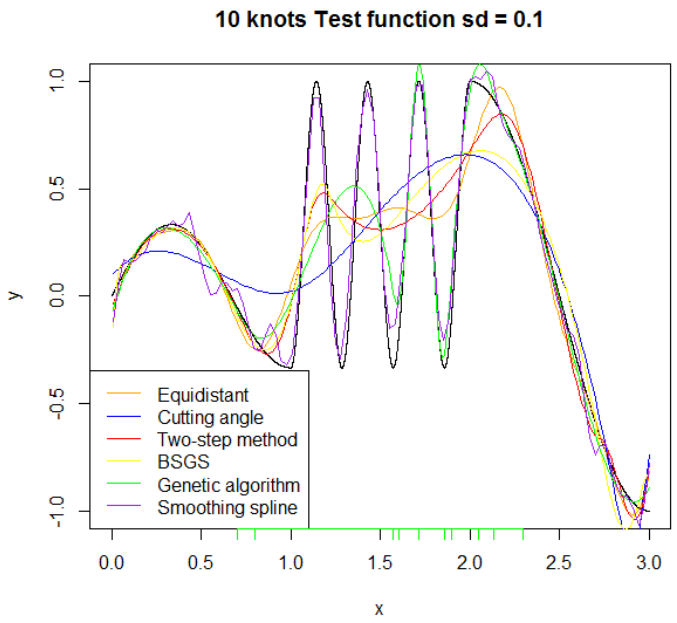
is an optimal amount of knots to use.
With regard to the performance of the different methods compared to each other, still the genetic algorithm seems to perform the best. This is especially the case for the lower amount of knots, as this method gives the lowest MSE for 10 knots.
As was done for the Doppler function, it is also interesting to look at the RSS for the Test function. These results can be found in Table 6.

Table 6 The RSS of the different methods on data samples from the Test function ($\sigma = 0.1$).

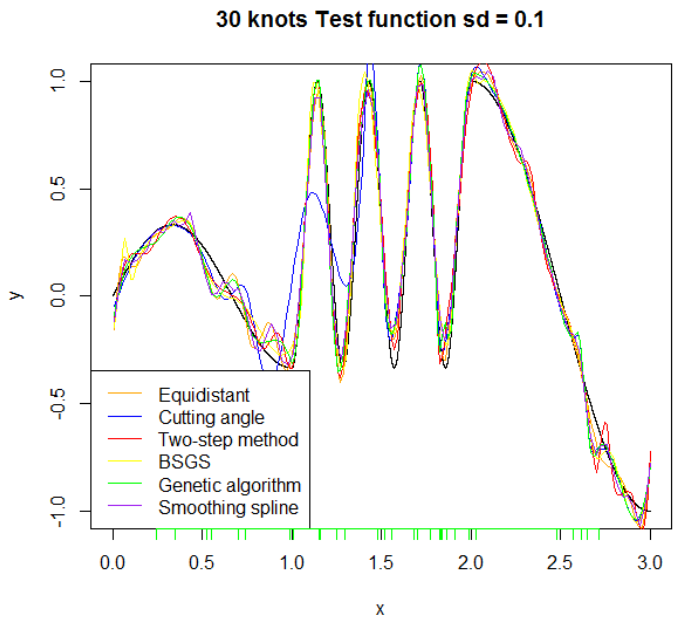
Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0833	0.0053	0.0073	0.0013	0.0027	0.0007
Cutting angle	0.0909	0.0128	0.0286	0.0180	0.0036	0.0026
Two-step method	0.0837	0.0065	0.0087	0.0038	0.0023	0.0006
BSGS	0.0864	0.0110	0.0265	0.0172	0.0032	0.0013
Genetic algorithm	0.0525	0.0045	0.0060	0.0012	0.0016	0.0004
Smoothing spline	mean		sd			
	0.0045		0.0013			

Comparing the results in Table 6 to the results in Table 3 shows that it is similar in comparison to the MSE for both functions. Still the same methods are performing better with regard to the RSS as the methods performing with regard to the MSE. This means that the RSS seems to be a good substitute to test which method performs best. This can be supported by the visual results. These can be found in Figure 9.

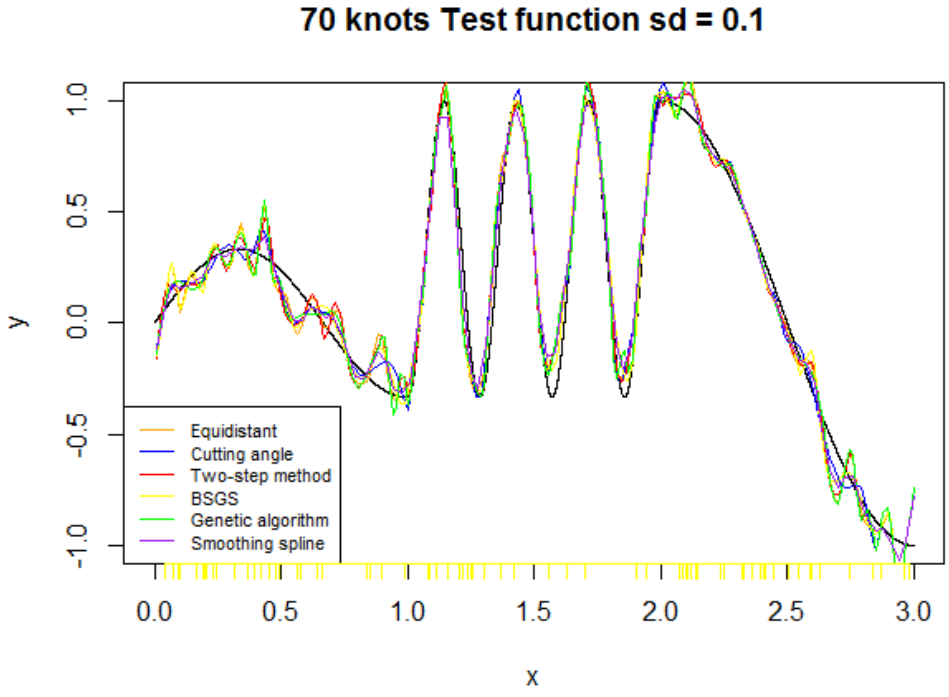
Looking at the visual results from Figure 9 it can be seen that some of the same problems occur as in the visual results from Figure 7. Looking at the results for 10 knots in Figure 9a it is seen that the knot-placing methods are especially struggling to find a proper estimation of the original function in the interval $[1, 2]$. Looking at the knot placement of the genetic algorithm, it can be seen that in this interval especially the part where no knots are placed is estimated worse. In the interval $[1.5, 2]$ some knots are placed, which results in quite a good estimation of the original function by the genetic algorithm. This algorithm does not output any knots in the intervals $[0, 0.7]$ and $[2.3, 3]$, but looking at the estimation of the original function, this is still quite good. This leads to the conclusion that these intervals do not require a lot of knots to be placed here.
The estimation of the interval $[1, 2]$ is already much improved when 30 knots are used, and an almost perfect resemblance is achieved for all methods if 70 knots are used. As this is the more fluctuating part of the function, this supports what was found when looking at the results from Figure 7, as the more fluctuating parts are better estimated if more knots are used. However, in both Figures 9b and 9c it can also be seen that the estimations by the splines are fluctuating too much in the other parts of the interval if more knots are used. Especially in Figure 9c it can be seen that a lot of knots are placed by the BSGS method in the interval $[0, 0.5]$. However, it can also be seen that the data is overfitted in this interval, so in this interval less knots should be used. This supports the conclusion that there is a trade-off between estimating the rougher or smoother parts of the original function better. Therefore it is likely that there can be found an optimal amount of knots to use for each different data sample and knot-placing method.
Besides the visual results, it is also still interesting to look at the computation time for the Test function to see whether this might be influenced by the type of data sample. The computation times can be found in Table 7.



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 9: Comparing the methods on Test function data samples ($\sigma = 0.1$).

Table 7 The computation time (in seconds) of the different methods on data samples from the Test function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0001	0.0007	0.0002	0.0017	0.0006	0.0033
Cutting angle	4.748	0.3315	5.399	0.5654	6.737	0.4025
Two-step method	2.459	0.2302	15.611	1.967	91.337	2.720
BSGS	0.7675	0.0631	1.279	0.0832	2.78	0.1438
Genetic algorithm	17.360	0.8253	31.116	1.086	70.409	1.903
Smoothing spline	mean			sd		
	0.0014			0.0047		

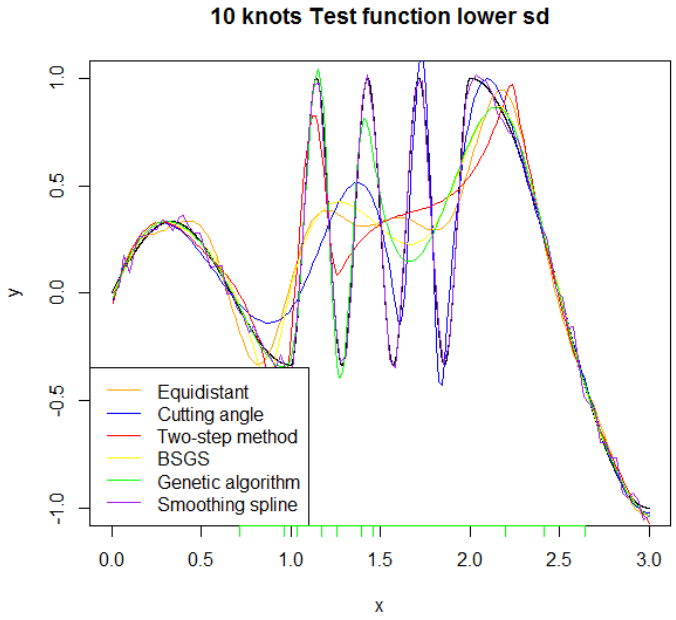
Comparing the results with regard to the computation time in Table 7 to the computation times in Table 4 show no significant differences for the Cutting angle method and the BSGS method. For the Two-step method and the genetic algorithm, there can be seen some differences with regard to the computation time.
For the Two-step method, looking at 10 and at 70 knots, it can be seen that the mean computation time for the Test function is quite a bit lower than for the Doppler function. However, taking a closer look at the standard deviation here shows that this is also quite a lot larger for the Doppler function as well. This indicates that it is likely that the methods perform quite similar in general, but that in the Doppler function some data samples took quite some time to process for these methods, which causes the mean to increase as well.
For the genetic algorithm, looking at 10 and at 30 knots, it can be seen that the mean computation time is larger for the Test function than it is for the Doppler function. These differences disappear if the amount of knots is equal to 70. However, the difference for the lower amounts of knots is significant, so the computation time of this method is probably dependent on the data set for a lower amount of knots. For the other methods, it is, despite some differences for the Two-step method, safe to say that the computation time is only dependent on the amount of knots available and not on the data sample that is used.
At last, for this function it is also interesting to take a look at what happens when the noise standard deviation of the sample is lower. Again, for the purpose of readability the full results are not included here and the most interesting results are explained. Full results for the lower noise standard deviation $\sigma = \sqrt{0.001}$ can be found in Appendix B.2.
Like for the aforementioned Doppler function, looking at the different MSE's for the Test function again shows a lot of similarities if the noise standard deviation is varied. The most notable similarities are that the same methods that were worse for especially 30 knots, still are the ones that are worse if the noise standard deviation is lower. Furthermore, there is also a large improvement with regard to the MSE from using 10 knots to using 30 knots, as was also seen in Table 5. However, unlike with noise standard deviation $\sigma = 0.1$, the lower noise standard deviation does not show a stagnation in improving MSE for 70 knots. This is similar to what was seen for the Doppler function.
Furthermore, it is interesting to look at the RSS for the lower noise standard deviation on the Test function as well. For the results with regard to the RSS there are a lot of similarities to the results that were explained regarding the MSE. Again the results do not show any differences with regard to which method is performing best. As in Table 6 it was not seen that the results for 30 and for 70 knots were quite similar, it is no surprise that this similarity is also not seen for the lower noise standard deviation. The only remarkable difference between the RSS and the MSE with regard to the lower noise standard deviation is the magnitude in which a lower noise standard deviation leads to a lower MSE and RSS, respectively. Where the difference with regard to the MSE was rather small, for the RSS the difference is significantly larger. This means that a lower noise standard deviation influences the RSS more than it influences the MSE. This can be explained by the fact that a lower noise standard deviation results in data points that are likely closer to the original function, which most of the time also makes it easier to estimate them.

We will also investigate the visual aspects of the Test function sampled with a lower noise standard deviation. These results can be seen in Figure 10.

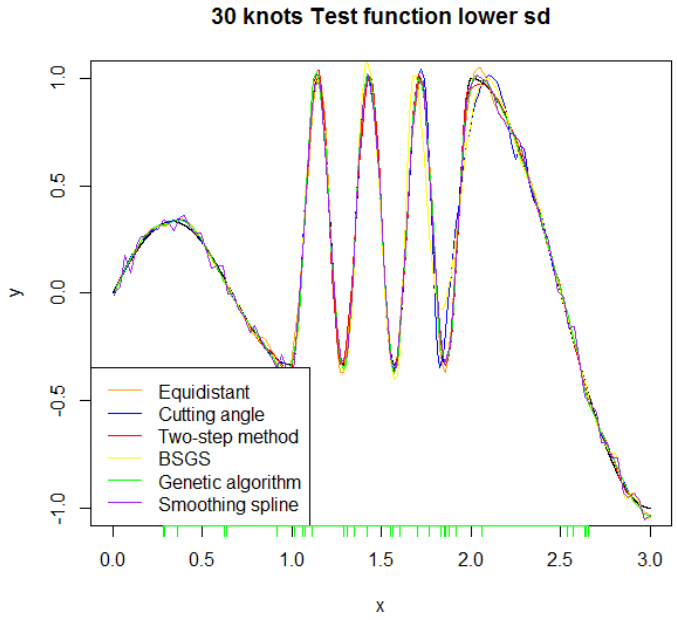
Comparing the results from Figure 10 to Figure 9 shows some interesting things. First of all, most of the methods perform a lot better if the noise standard deviation is smaller. This is best seen when comparing Figure 9a to Figure 10a. In this comparison, it can be seen that the Cutting Angle method, the BSGS method and the genetic algorithm all perform better in Figure 10a. Taking a closer look at the genetic algorithm in Figure 10a and its knot placement also supports a conclusion drawn from Figure 9a. In Figure 9a the knots in interval $[1.5, 2]$ enabled a good estimation of the original function here, while the lack of knots in interval $[1, 1.5]$ disables a good estimation here. In Figure 10a it is the other way around, as there are knots present in the interval $[1, 1.5]$ and there is a lack of knots in the interval $[1.5, 2]$. The same thing happens as for the higher noise standard deviation, as the estimating spline resembles the original function quite well if there are knots placed, and quite poorly if there is a lack of knots in a rougher part of the original function.

The trend that methods perform better for a lower noise standard deviation continues for 30 knots, as the BSGS method and the Cutting Angle method perform better in Figure 10b than they do in Figure 9b. For 70 knots it also seems to be that the methods perform better if the noise standard deviation is lower. This can especially be seen when comparing the intervals $[0, 1]$ and $[2.6, 3]$ in Figures 9c and 10c. In these intervals, it can be seen that using the higher noise standard deviation leads to a more wiggly estimation of the spline, while the original function is quite smooth. This leads to the impression that having a lower noise standard deviation leads to a smoother estimation. This does, however, not mean that the estimation is relatively better, as it still poses the problem that the data seems to be interpolated in the mentioned intervals. This is not preferable, and thus the idea that there is an optimal amount of knots is supported by this notion. Because of the differences between the different noise standard deviations it is likely that this optimal amount is dependent on the noise standard deviation of the data sample.

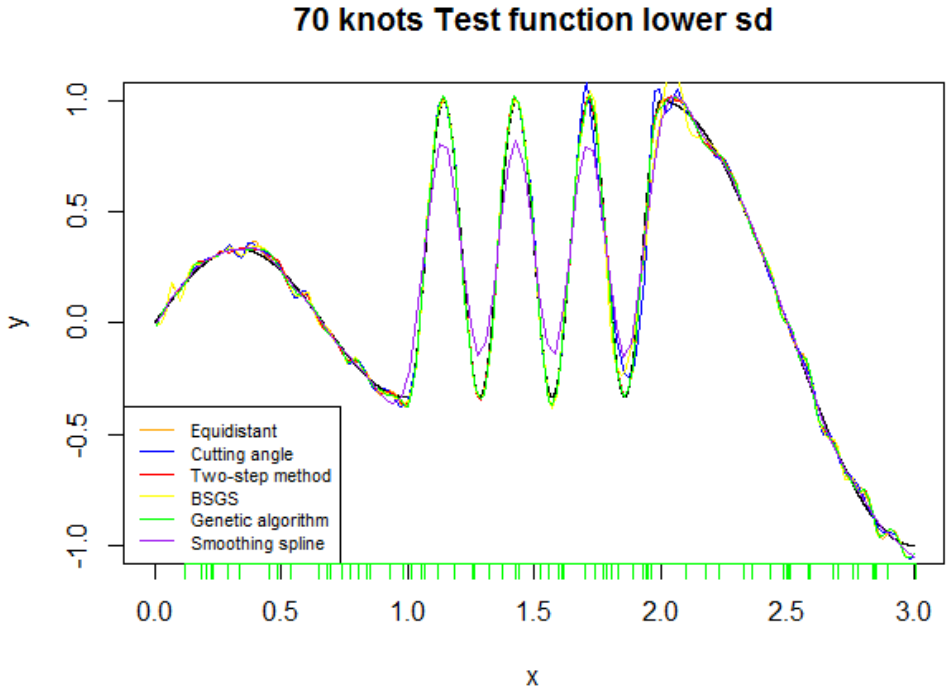
At last, it is for the Test function also interesting to investigate whether the noise standard deviation influences the computation time. Comparing the results from the lower noise standard deviation with the results from Table 7 show no significant differences. This is in correspondence with what was seen for the Doppler function. This also supports the impression that the computation time is not dependent on the data, but rather on the amount of knots used to estimate this data.



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 10: Comparing the methods on Test function data samples $\sigma = \sqrt{0.001}$.

5 Practical use on data sets

In this chapter, the methods are compared for the two pre-defined data sets for which the original function is unknown. These are the Titanium Heat data set and the Simulated Motorcycle Accident data set. This is done to see how well methods perform if the original function is unknown. First the data sets are described, after which the comparison is done. For these methods, the comparison can be done with regard to the computational time, the RSS and the visual aspects of the estimating spline.

5.1 Data sets

In this chapter we investigate the pre-defined data sets. The first one is the Titanium Heat data set, which can be seen in Figure 11.

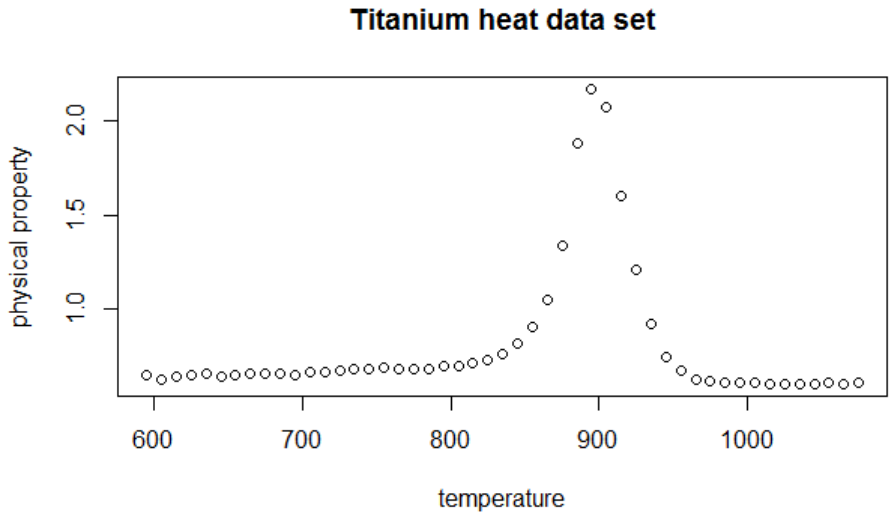


Figure 11: The Titanium Heat data.

This data set is chosen as it is considered one of the standard tests for data fitting by splines, since it is proven to be difficult to estimate using classical techniques and the amount of noise is significant present. The data set represents a thermal property of titanium with regard to the temperature. The data set consists of 49 observations. This data set thus does not have a large amount of data points, while the interval which the data set is in is quite large, so this might also give some interesting insights in the behaviour of the different methods.
The second data set that is investigated is a data set that gives a series of measurements of head acceleration from a simulated motorcycle accident, used to test crash helmets. This data set can be seen in Figure 12.

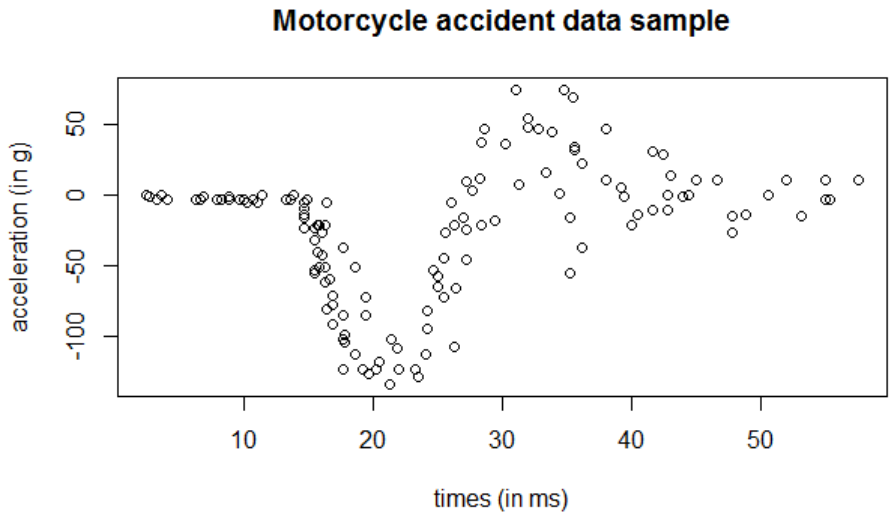


Figure 12: Head movement during a simulated motorcycle accident.

This data set is chosen as it shows deviations over time with regard to the behaviour of the head acceleration. The data set consists of 133 observations, spread unevenly over the time that is covered by the data set. As this data set has several different observations at the same time, the data set is modified to be able to perform the smoothing spline method. This is done by taking the average value for all unique times that are covered in the data set. This results in a new data set consisting of 94 observations.

5.2 Results

In this chapter the two data sets that have not been sampled from an original function are evaluated. This is done for each data set individually and for all different amounts of knots and methods. For these data sets, the RSS, computation time and the visual results are evaluated.

5.2.1 Titanium Heat data

The Titanium Heat data set is used in multiple spline studies to test whether a knot placing method is accurate. Therefore it is interesting to look at this data set in this report as well. As this is a data set for which the original function is unknown, the MSE can not be computed. Also it is only one data sample, so the estimating splines are only computed once using each method, as it is expected that the methods output the same results for the same data set. Therefore the results do not constitute of a mean and a standard deviation, but just of the results of this one run. The results with regard to the RSS can be found in Table 8.

Table 8 The RSS of the different methods on the Titanium Heat data set.

Method	10 knots	20 knots	35 knots
Equidistant	0.0037	$2.830 \cdot 10^{-4}$	$2.652 \cdot 10^{-5}$
Cutting angle	0.0240	0.0094	0.0002
Two-step method	0.0333	0.0093	0.0032
BSGS	0.0055	0.0026	0.0001
Genetic algorithm	0.0523	0.0047	0.0047
Smoothing spline	$1.243 \cdot 10^{-25}$		

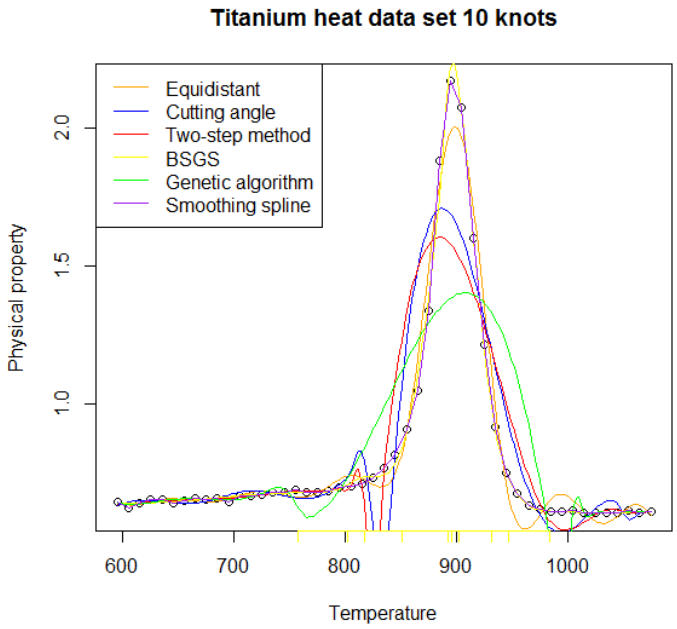
Looking at these results, it can be seen that the RSS decreases as the amount of knots grows for all methods. This means that more knots would lead to a more accurate representation of the data points with regard to the RSS. The genetic algorithm shows the most improvement as the amount of knots grows, but the BSGS method still outperforms it by quite a lot in every case. This is remarkable, as we have not seen this happening for the previous functions, so perhaps this method might perform better for other specific cases as well.
To investigate the results from this data set further, we will look at the visual results of the knot-placing methods. These results can be found in Figure 13.

Looking at Figure 13, we see that all methods seem to improve on their performance if the amount of knots are higher. Unlike in the case of the Doppler function for instance, it does not happen that the closer approximation near the rougher parts leads to a rougher approximation at the smoother parts. This might be due to the characteristics of this data set, which is resembling a single spike on a flat line. This data set seems to be best approximated using the BSGS method. Looking at the knot placement of the BSGS method in Figure 13a it can be seen that the knots are concentrated around the spike of the data set. This leads to a quite accurate approximation of this spike, and it gives the impression that more knots should be placed here to get an accurate estimation of this data set. The BSGS method was also the fastest performing method in chapter 4. Therefore we want to take a look at the computation times whether this is still the case. The computation times can be found in Table 9.

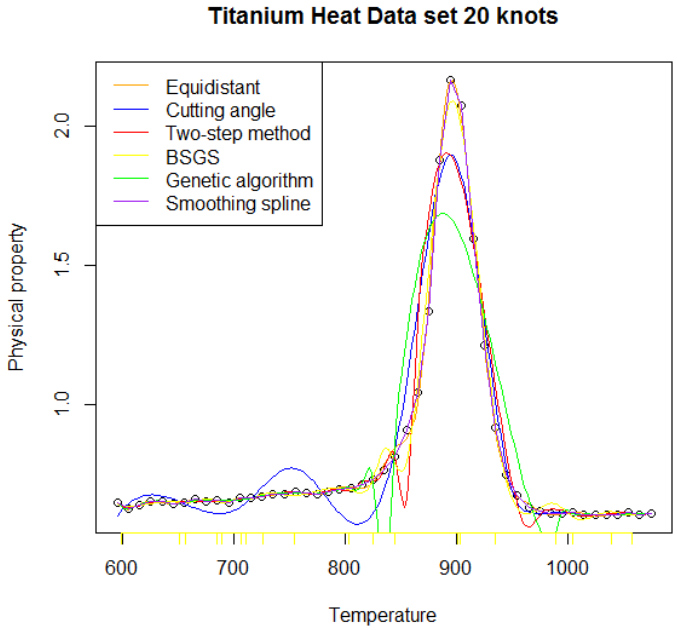
Table 9 The computation time (in seconds) of the different methods on the Titanium Heat data set.

Method	10 knots	20 knots	35 knots
Equidistant	0	0	0.02
Cutting angle	5.63	5.46	5.89
Two-step method	2.66	7.63	19.79
BSGS	0.64	0.83	1.16
Genetic algorithm	15.07	20.44	29.22
Smoothing spline	0		

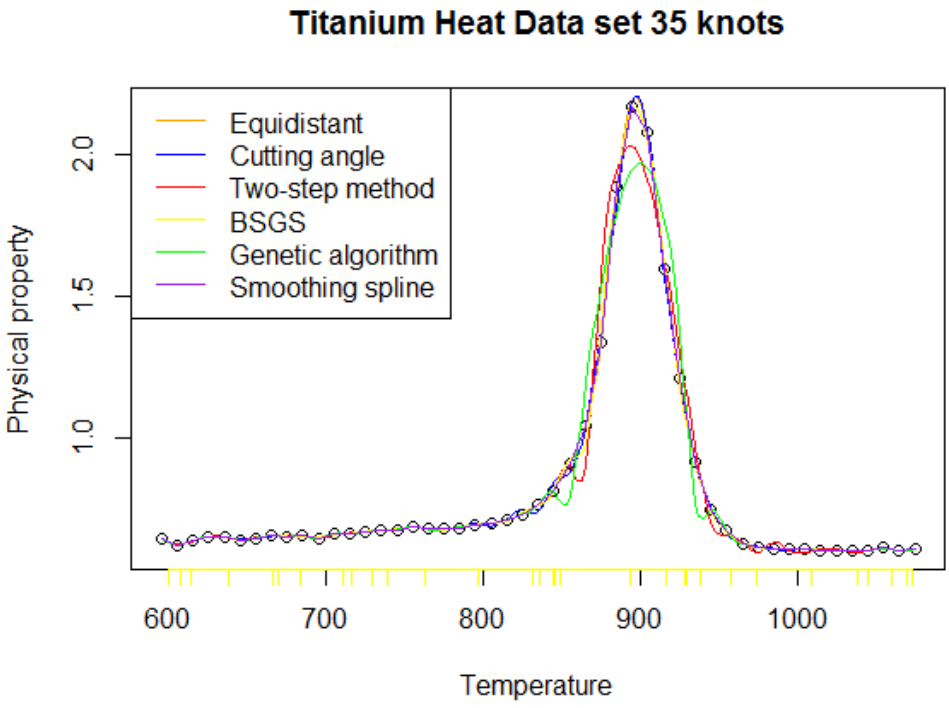
The results in Table 9 do not show any differences from the results in Table 4, if you take into account the different values for the amount of knots. This means that the computation time of the different methods is not dependent on the type of data provided or the amount of data provided, but it is dependent on the amount of knots that can be used. It can be concluded that this data set can quite accurately be approximated using quite a few knots, if they are placed at the correct locations. Looking at this data set supports the conclusion that more knots should be placed at the rougher parts of the data set. The characteristics of this data set lead to the conclusion that the BSGS method performs best for this data set, and it is also the fastest method. Therefore we can conclude that this method might be interesting to try for specific data sets, as it leads to a good estimation in this case.



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 20 knots.



(c) The different methods compared to each other for 35 knots.

Figure 13: Comparing the methods for the Titanium Heat data set.

5.2.2 Motorcycle accident

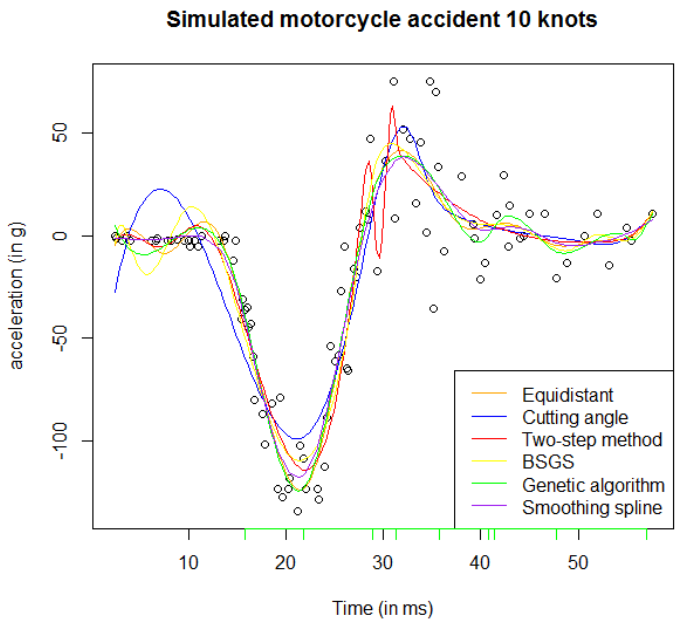
In this chapter the Simulated Motorcycle Accident data set will be investigated. This data also consists of a single data set with an unknown original function, thus the MSE can not be computed and we only take one computation of each method into consideration. However, the data set does show some differences with the Titanium Heat data, as the data appears a bit more scattered and the original function is therefore probably more difficult to find. For this data set, we will also first investigate the RSS, for which the results can be found in Table 10.

Table 10 The RSS of the different methods on the Simulated Motorcycle Accident data set.

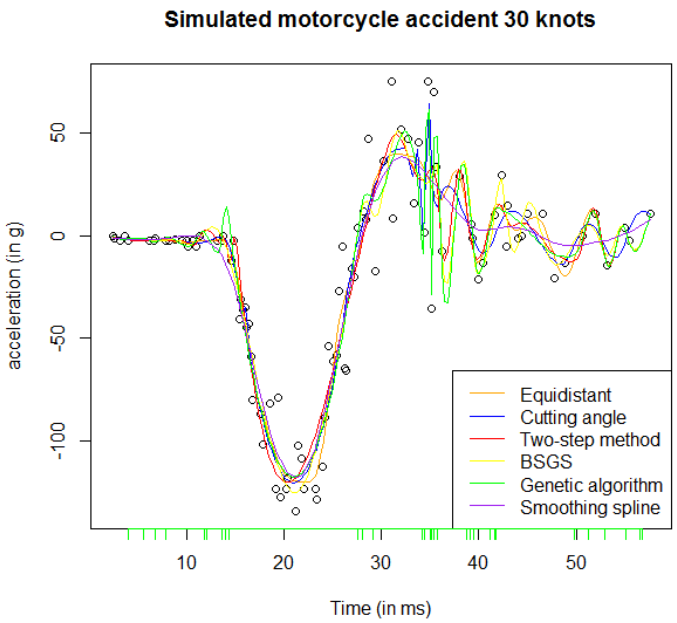
Method	10 knots	30 knots	70 knots
Equidistant	313.72	271.85	177.57
Cutting angle	523.09	270.64	192.58
Two-step method	345.99	293.02	171.19
BSGS	382.98	261.83	99.28
Genetic algorithm	308.52	197.92	36.43
Smoothing spline	313.64		

The results in Table 10 are of a different magnitude than for instance the results from Table 8, as the domain of the data is also a lot larger. Regardless of this matter, it still shows that the methods perform better with regard to the RSS if the amount of knots increases. This holds for all the different methods. Also, like in Table 3, the genetic algorithm outperforms the other methods by quite a lot. This is especially the case if the amount of knots is high. This leads to the impression that the performance of this method is barely dependent on the type of data that is used as input. This causes that the genetic algorithm feels reliable, as it performs much better than other methods for most of the data sets.
Another remarkable thing that we did not see before is the fact that all methods outperform the smoothing spline with regard to the RSS, even if only 30 knots are used. This didn't happen in any previous case and it is known that the smoothing spline minimizes the penalized RSS. This leads to the impression that the data set not only appears to look more scattered, but it also is as the penalty term on the smoothness of the spline seems to contribute a lot more respectively to the penalized RSS.
As the methods seem to perform better if the amount of knots increases, it is interesting to look at the visual results to see if this is supported by them. The visual results can be seen in Figure 14.

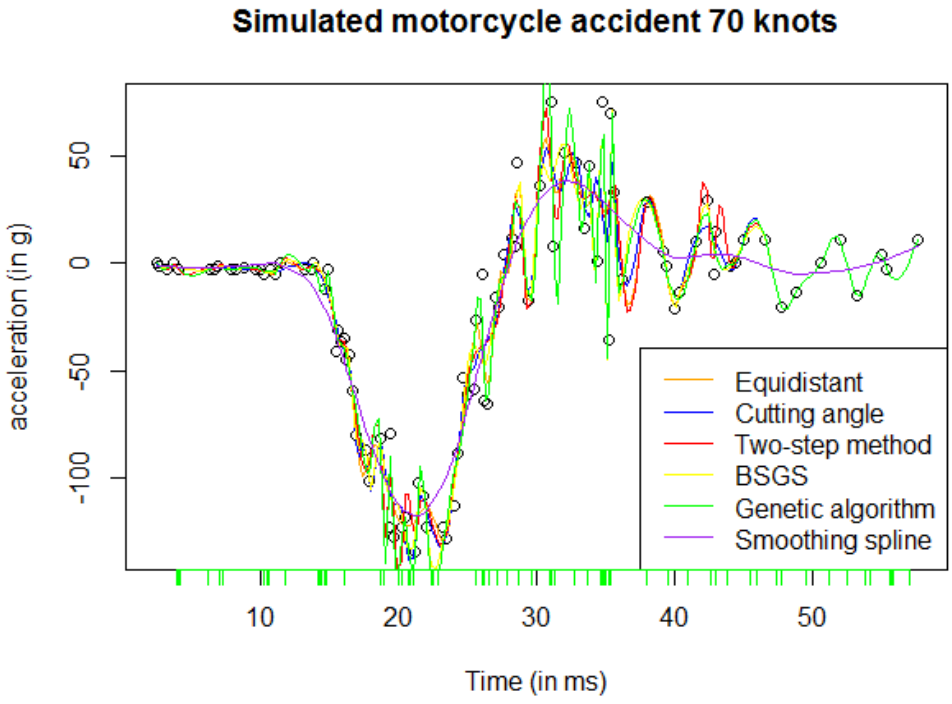
The results in Figure 14 show that having more knots does not necessarily mean that the trend of the data is covered better. Although the RSS is lower, this only means that the data points are approached more accurately, but this does not mean that the smoothness of the estimating spline is maintained. This statement is particularly supported by the visual results of the genetic algorithm. It can be seen that the genetic algorithm is still quite smooth in Figure 14a, but it already captures too many data points accurately in Figure 14b. This means that this is probably already too many knots for this method in combination with this data set, because the smoothness is not prevailed. Looking at the knot placements of the genetic algorithm, it can be seen in Figure 14a that there are no knots in the part before the impact. It still seems to be estimated quite accurately here, regardless of the lack of knots here. Finally looking at the genetic algorithm in Figure 14c, it shows that in this case the data points are almost simply interpolated. For all methods in Figure 14c it seems to be the case that there are too many knots used. This leads to the conclusion that a large amount of knots is not preferable if the characteristics of the data seems to be with a very high noise standard deviation.
At last it is interesting to see what happens to the computation time if the domain of the data is bigger. These results can be found in Table 11.



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 14: Comparing the methods on the Simulated Motorcycle Accident data set.

Table 11 The computation time (in seconds) of the different methods on the Simulated Motorcycle Accident data set.

Method	10 knots	30 knots	70 knots
Equidistant	0.01	0	0
Cutting angle	6.34	6.78	7.42
Two-step method	3.67	17.7	96.3
BSGS	0.75	1.25	2.61
Genetic algorithm	17.25	30.78	68.57
Smoothing spline	0		

The results in Table 11 show a lot of similarities to the results in Table 4. It can still be seen that the computation time steadily increases if the amount of knots increases. However, all computation times in Table 11 are slightly higher than the computation times in Table 4, which shows that the computation time is also slightly dependent on the domain of the data set.

6 Conclusion

Evaluation of the results in chapter 4 and chapter 5 can be used to draw some conclusions regarding the methods that are described in this report and when to use them:

- From the results in chapter 4, it can be concluded that the genetic algorithm [10] performs best in general. This is seen by the evaluation of the MSE and the graphical representation of the different cases. Chapter 5 supports this conclusion, as this method also has a quite accurate approximation in this chapter with regard to the RSS and based on the graphical results.

- From chapter 5, it is found that the BSGS method [10] performs quite good if the characteristics of the data set is favourable for this method to use. This method is also the fastest method to compute from all the investigated methods. Therefore, it is useful to compute this method in quite some cases, as it already gives a quite accurate representation, while it is fast to compute as well.

- From the results in chapter 4 and 5 it can be concluded that using more knots does not always have a significant improvement on the accuracy of the estimating spline. Having more knots might actually lead to a more inaccurate estimation, as the risk of interpolating the data points becomes higher if the amount of knots used is higher. Therefore it leads to the conclusion that each method and data set has an optimal amount of knots to be used for which the trade-off between the smoothness of the estimating spline and the accuracy of this spline is optimal. Finding the optimal amount of knots to be used is left as future work.

- Chapter 5 shows that the computation time of all methods increase as the amount of knots used increases. However, some methods are more vulnerable with regard to computation time than others. For instance, the genetic algorithm [10] is in general slower than the other methods for 10 and 30 knots, but for 70 knots the Two-step method [9] is the slowest. This shows that the Two-step method [9] is very vulnerable for the computation time to increase as the amount of knots increases. Therefore, this method is not preferable to use if the amount of knots that should be used is large. Methods that are not very much influenced by the amount of knots that are used, are the Cutting angle method [7] and the BSGS method [10]. This leads to the conclusion that these methods are preferable to use if the amount of knots to be used is large.

- From chapters 4 and 5 it can be concluded that for most methods the computation time is independent of the data set that is used. The only exception that was found regarding this was the genetic algorithm [10]. This method shows a small dependency on the data set to be estimated with regard to the computation time of the method.

- From chapter 4 it can be concluded that a higher noise standard deviation results in a higher mean squared error. Also it can be concluded that the improvement of methods with regard to the MSE if knots are added is not influenced by the noise standard deviation. The variation of the noise standard deviation also does not influence which method performs the best.

- From chapter 4 it can be concluded that the RSS is a good substitute in selecting which method performs best. The value of the RSS is not representative for the value of the MSE, but it can be seen that the method that leads to the lowest MSE is also the method that leads to the lowest RSS. Therefore selecting the method that gives the lowest RSS if the original function is unknown very likely also leads to the method that gives the best estimation of the original function.

All of the conclusions above tell us that the genetic algorithm [10] performs best in the instances that are investigated in this report. From the characteristics that are found in this report, it can also be concluded that there might be instances for which a different method is desirable. It can

also be concluded that the RSS is a good substitute for the MSE to determine which method performs best.

7 Discussion

This report tries to give a nice overview of the performances of different methods with regard to spline fitting of data. Some things in this report could have been done better, which are:

- The different methods are only compared to each other based on 2 functions that data was sampled from and 2 other data sets. Therefore there is a nice overview of what the performances of the different methods are when those functions are used. However, to have a more general comparison of performances, it is interesting to make this comparison for multiple different functions. It could now still be possible that the best performing method for the data sets in this report performs worst for several different other functions.
- The different methods are only compared to each with regard to 3 different amounts of knots. It could be interesting to also investigate what happens if for instance only 5 knots are used, or what happens if the amount of knots used is equal to the amount of data points. Adding more different amounts would have led to a more reliable conclusion and possibly also new insights regarding the performance compared to the equidistant knots or the smoothing spline with knots at the data points.
- The methods proposed concerning Gauss-Newton optimization with Marquardt modification [5] and the Bisecting method [8] are not included if the results are discussed. This is due to the fact that there are expected to be errors in the coding of these methods. After an extensive search in the code these errors could not be spotted. It would have been better if these methods were coded properly, as now there can be no conclusions drawn with regard to them, while they might be interesting to investigate further.
- This report tries to give an insight to what the influence of the noise standard deviation of the sample is on each method. This is done by decreasing it with a factor $\sqrt{10}$. To have a more reliable conclusion this could have been done for more different values. This was not done due to the fact that this is very time-consuming.
- This report focuses on how the different methods perform when the amount of knots is fixed. Besides this, it is also interesting to investigate methods that determine the optimal amount of knots that should be used for each data set. Also the investigation on combining this with a method that optimizes the knot placement is interesting and unfortunately not present in this report.

Besides these flaws that are still in the report, it can however be very useful for future work on this subject. The flaws that were mentioned above could of course be tried to be fixed. This is very likely to give a better insight in the performance of the different methods in general.
Any of the future work regarding applying the methods to more different functions and/or with more different noise standard deviations is recommended to be done on a more professional computer. The computer on which this research is done is not the fastest, as the computation times already show for some methods. Therefore it can be a very extensive job if a lot of information is tried to be gained for these methods, so it is beneficial if this is done on a fast computer.
Furthermore, it is recommended to determine beforehand in which language the code is implemented. In this report mostly R is used, but it could also be very much possible that programming in Matlab is faster or gives better results.
At last, in the future it is also interesting to see whether implementing a combination of methods would lead to meaningful results. This also probably poses new challenges, such as determining at which point the one method is replaced by the other. Also determining which methods should be combined will give some challenges, but combining methods might lead to better results, which is interesting to investigate in the future. All in all, this report gives a nice basic overview of different methods regarding spline fitting to data. This overview consists of the basic properties of each method and an explanation which mathematical methods each method uses to obtain the knot vector. Besides this overview also a small comparison is given between the methods, which can be extended in the future.

References

[1] L. Metcalf, W. Casey, "Chapter 4- Introduction to data analysis," in *Cybersecurity and Applied Mathematics*, B. Romer and A. Valutkevich, British Library: Elsevier, 2016, pp. 43-65. Available: https://doi.org/10.1016/B978-0-12-804452-0.00004-X

[2] B. Ikenaga (2017), *Linear Independence* [Online]. Available: http://sites.millersville.edu/bikenaga/linear-algebra/linear-independence/linear-independence.html

[3] L. Wasserman, "All of Nonparametric Statistics", G. Casella, S. Fienberg and I. Olkin, Carnegie Mellon University: Springer-Verlag New York, 2006, pp. 1-11. Available: https://doi.org/10.1007/0-387-30623-4

[4] L. Schumaker, "Spline Functions: Basic Theory", 3^{rd} edition, Cambridge: Cambridge University Press, 2007.

[5] D. L. B. Jupp, "Approximation to Data by Splines with Free Knots", *SIAM Journal on Numerical Analysis*, vol. 15, No. 2, pp. 328-343, Apr 1978. Available: https://www.jstor.org/stable/2156756

[6] G. H. Golub, V. Pereyra, "The differentiation of Pseudo-Inverses and Nonlinear Squares Problems Whose Variables Separate", *SIAM Journal on Numerical Analysis*, vol. 10, Issue 2, pp. 413-432, 1973. Available: https://doi.org/10.1137/0710036

[7] G. Beliakov, "Least squares splines with free knots: global optimization approach", *Applied Mathematics and Computation*, vol. 149, issue 3, pp. 783-798, Feb 2004. Available: https://doi.org/10.1016/S0096-3003(03)00179-6

[8] V. T. Dung, T. Tjahjowidodo (2017), "A direct method to solve optimal knots of B-spline curves: An application for non-uniform B-spline curves fitting", *PLoS ONE*, vol. 12, issue 3, e0173857. Available: https://doi.org/10.1371/journal.pone.0173857

[9] H. Kang, F. Chen, Y. Li, J. Deng, Z. Yang, "Knot calculation for spline fitting via sparse optimization", *Computer-Aided Design*, vol. 58, pp. 179-188, Jan 2015. Available: https://doi.org/10.1016/j.cad.2014.08.022

[10] S. Spiriti, R. Eubank, P.W. Smith, D. Young, "Knot selection for least-squares and penalized splines", *Journal of Statistical Computation and Simulation*, vol.83, No. 6, pp. 1020-1036, 2013. Available: https://doi.org/10.1080/00949655.2011.647317

A Doppler function

A.1 Noise standard deviation $\sigma = 0.1$.

A.1.1 MSE

Table 12 The MSE of the different methods on data samples from the Doppler function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0264	0.0005	0.0144	0.0008	0.0103	0.0013
Cutting angle	0.0357	0.0102	0.0149	0.0054	0.0112	0.0026
Two-step method	0.0259	0.0067	0.0116	0.0023	0.0111	0.0015
BSGS	0.0347	0.0097	0.0158	0.0052	0.0174	0.0838
Genetic algorithm	0.0133	0.0014	0.0073	0.0014	0.0106	0.0014
Smoothing spline	mean			sd		
	0.0098			0.0018		

A.1.2 RSS

Table 13 The RSS of the different methods on data samples from the Doppler function ($\sigma = 0.1$).

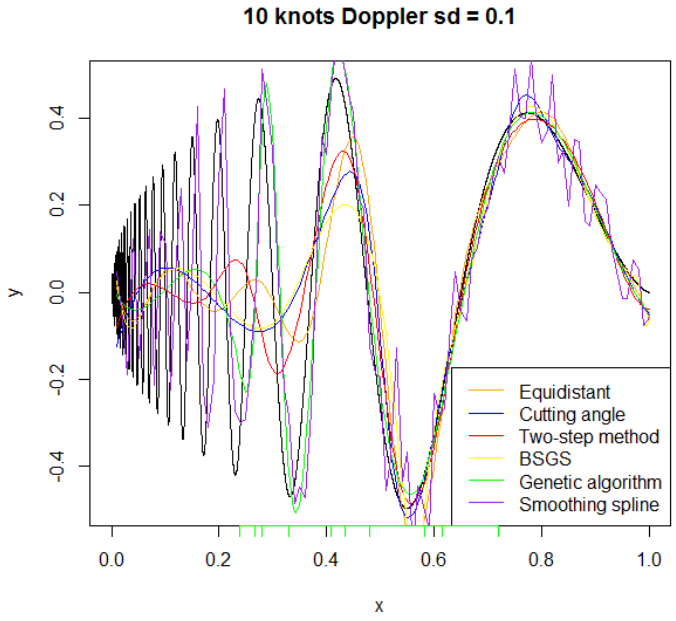
Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0320	0.0037	0.0161	0.0022	0.0030	0.0008
Cutting angle	0.0488	0.0114	0.0172	0.0056	0.0055	0.0025
Two-step method	0.0311	0.0071	0.0121	0.0028	0.0028	0.0010
BSGS	0.0404	0.0104	0.0174	0.0055	0.0154	0.1382
Genetic algorithm	0.0181	0.0025	0.0071	0.0012	0.0015	0.0005
Smoothing spline	mean			sd		
	0.0062			0.0047		

A.1.3 Computation time

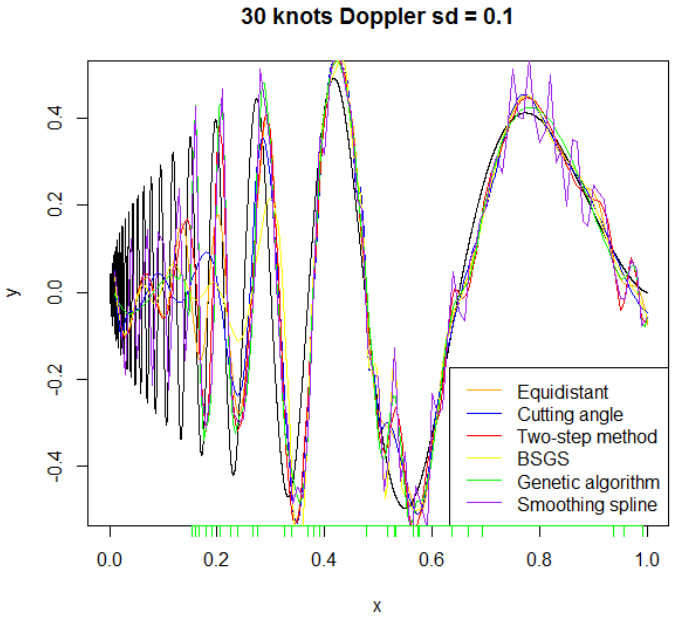
Table 14 The computation time (in seconds) of the different methods on data samples from the Doppler function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0008	0.0086	0.0002	0.0017	0.0004	0.0026
Cutting angle	4.853	0.3905	5.117	0.4003	6.013	0.4676
Two-step method	3.001	0.6389	17.405	1.581	91.424	5.988
BSGS	0.6438	0.0846	1.092	0.1147	2.798	0.1686
Genetic algorithm	14.574	1.768	26.862	2.735	71.203	2.452
Smoothing spline	mean			sd		
	0.0011			0.0043		

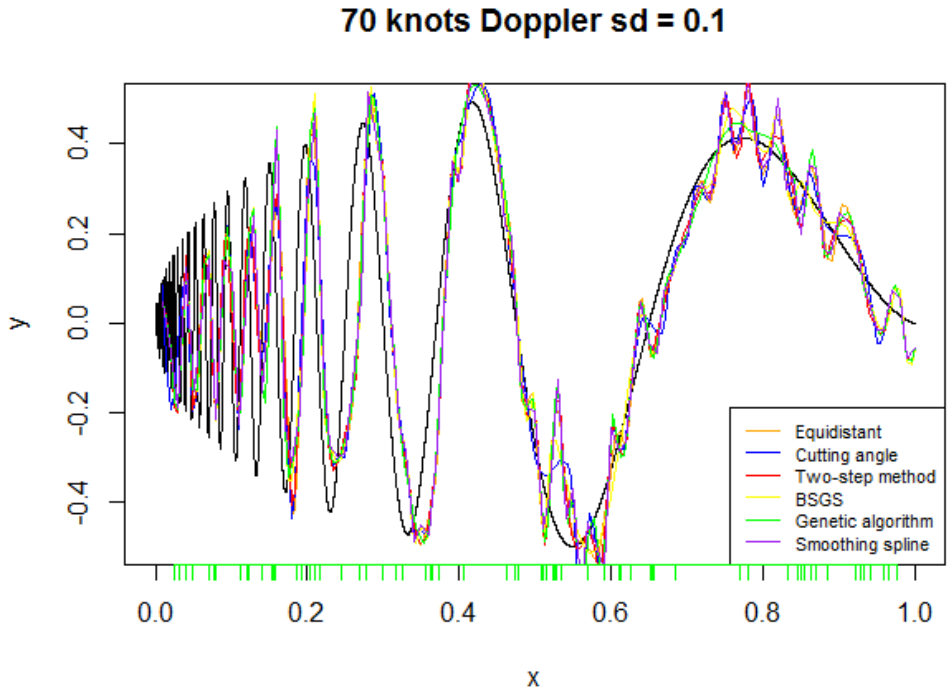
A.1.4 Visual results



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 15: Comparing the methods on the Doppler function data samples ($\sigma = 0.1$).

A.2 Noise standard deviation $\sigma = \sqrt{0.001}$.

A.2.1 MSE

Table 15 The MSE of the different methods on data samples from the Doppler function ($\sigma = \sqrt{0.001}$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0252	0.0001	0.0114	0.0001	0.0039	0.0002
Cutting angle	0.0342	0.0100	0.0122	0.0045	0.0052	0.0020
Two-step method	0.0220	0.0036	0.0072	0.0019	0.0044	0.0006
BSGS	0.0342	0.0090	0.0126	0.0050	0.0056	0.0021
Genetic algorithm	0.0126	0.0050	0.0031	0.0003	0.0036	0.0002
Smoothing spline	mean			sd		
	0.0038			0.0002		

A.2.2 RSS

Table 16 The RSS of the different methods on data samples from the Doppler function ($\sigma = \sqrt{0.001}$).

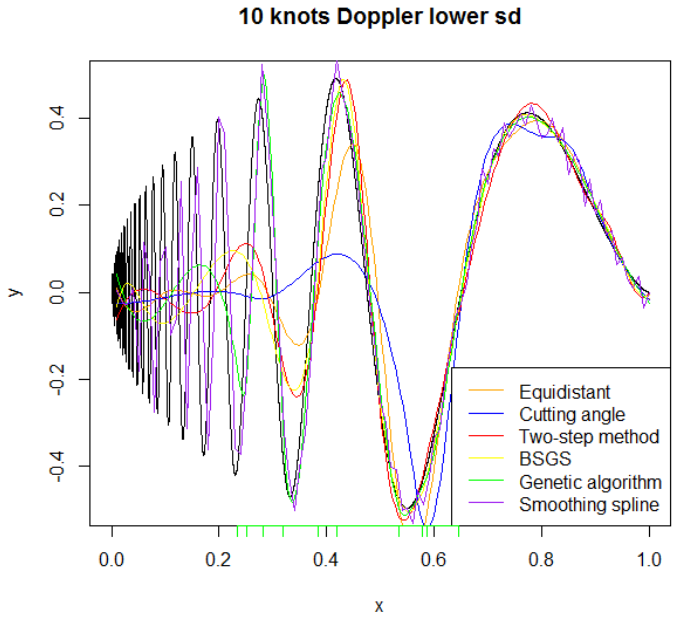
Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0245	0.0010	0.0103	0.0007	0.0007	0.0001
Cutting angle	0.0336	0.0100	0.0111	0.0047	0.0055	0.0025
Two-step method	0.0214	0.0036	0.0060	0.0019	0.0015	0.0007
BSGS	0.0335	0.0090	0.0114	0.0050	0.0030	0.0021
Genetic algorithm	0.0105	0.0008	0.0019	0.0003	0.0004	0.0001
Smoothing spline	mean			sd		
	$1.193 \cdot 10^{-18}$			$1.577 \cdot 10^{-19}$		

A.2.3 Computation time

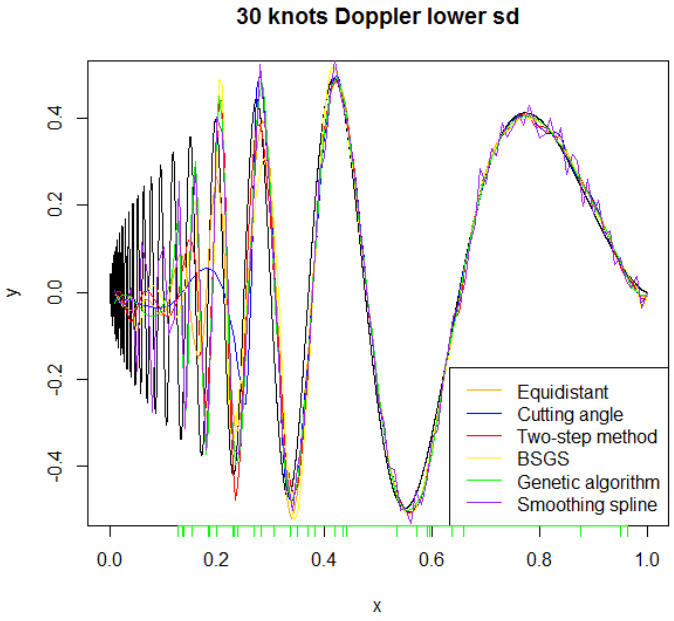
Table 17 The computation time (in seconds) of the different methods on data samples from the Doppler function ($\sigma = \sqrt{0.001}$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0	0	0	0	0.0004	0.0027
Cutting angle	4.787	0.3594	5.033	0.3742	5.936	0.4383
Two-step method	2.997	0.6234	17.410	1.582	91.071	5.965
BSGS	0.6413	0.0780	1.094	0.1164	2.801	0.1557
Genetic algorithm	14.59	1.700	26.890	2.639	71.219	2.295
Smoothing spline	mean			sd		
	0.0010			0.0040		

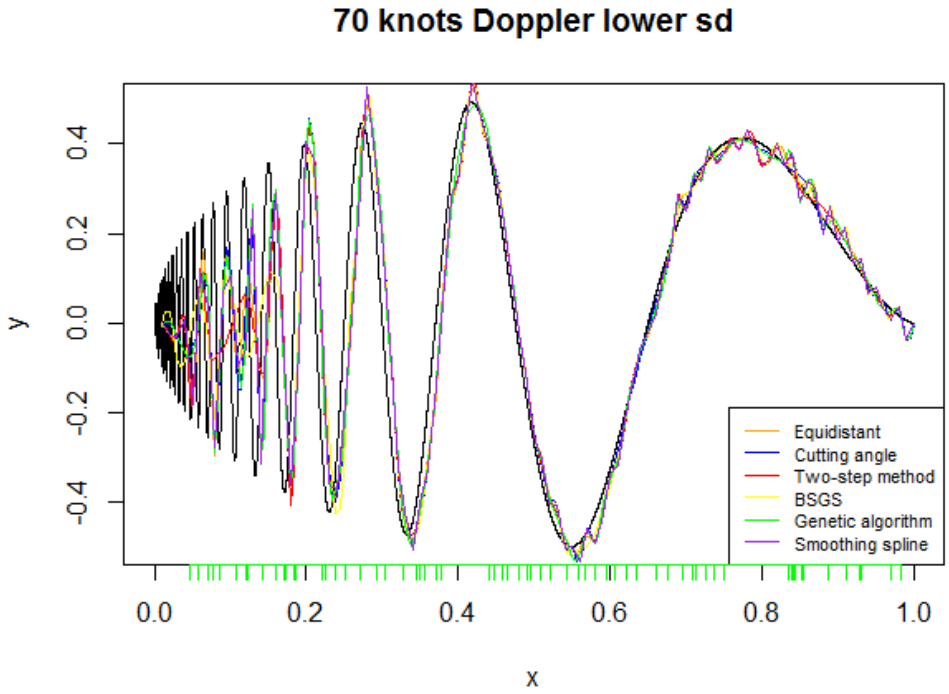
A.2.4 Visual results



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 16: Comparing the methods on the Doppler function data samples ($\sigma = \sqrt{0.001}$).

B Test function

B.1 Noise standard deviation $\sigma = 0.1$.

B.1.1 MSE

Table 18 The MSE of the different methods on data samples from the Test function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0769	0.0006	0.0050	0.0010	0.0080	0.0013
Cutting angle	0.0841	0.0119	0.0258	0.0178	0.0080	0.0027
Two-step method	0.0770	0.0040	0.0066	0.0032	0.0082	0.0014
BSGS	0.0801	0.0093	0.0242	0.0168	0.0080	0.0017
Genetic algorithm	0.0478	0.0025	0.0055	0.0012	0.0086	0.0013
Smoothing spline	mean			sd		
	0.0049			0.0011		

B.1.2 RSS

Table 19 The RSS of the different methods on data samples from the Test function ($\sigma = 0.1$).

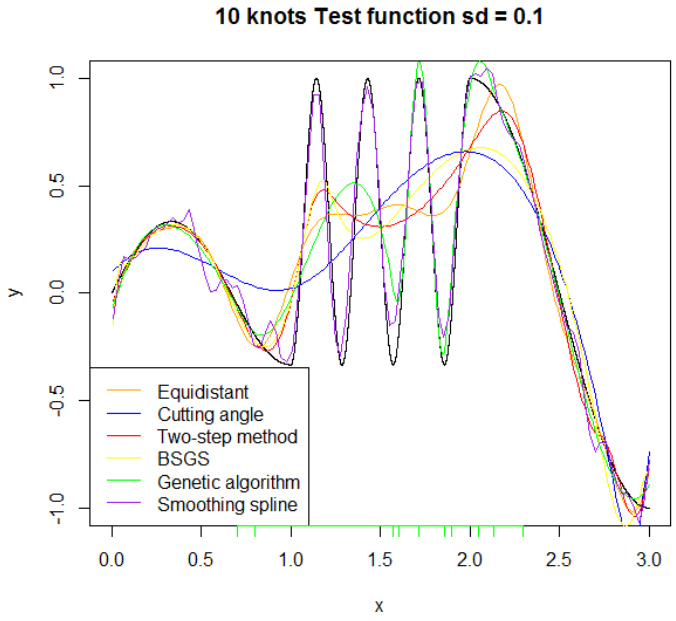
Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0833	0.0053	0.0073	0.0013	0.0027	0.0007
Cutting angle	0.0909	0.0128	0.0286	0.0180	0.0036	0.0026
Two-step method	0.0837	0.0065	0.0087	0.0038	0.0023	0.0006
BSGS	0.0864	0.0110	0.0265	0.0172	0.0032	0.0013
Genetic algorithm	0.0525	0.0045	0.0060	0.0012	0.0016	0.0004
Smoothing spline	mean			sd		
	0.0045			0.0013		

B.1.3 Computation time

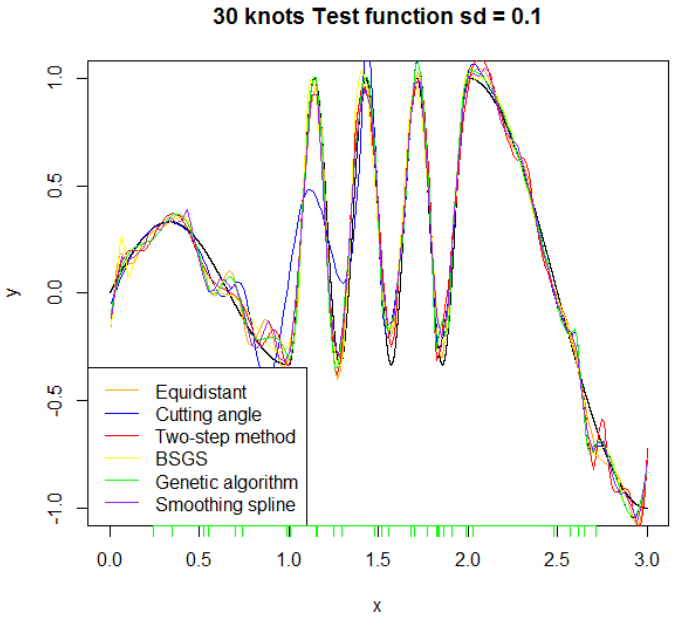
Table 20 The computation time (in seconds) of the different methods on data samples from the Test function ($\sigma = 0.1$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0001	0.0007	0.0002	0.0017	0.0006	0.0033
Cutting angle	4.748	0.3315	5.399	0.5654	6.737	0.4025
Two-step method	2.459	0.2302	15.611	1.967	91.337	2.720
BSGS	0.7675	0.0631	1.279	0.0832	2.78	0.1438
Genetic algorithm	17.360	0.8253	31.116	1.086	70.409	1.903
Smoothing spline	mean			sd		
	0.0014			0.0047		

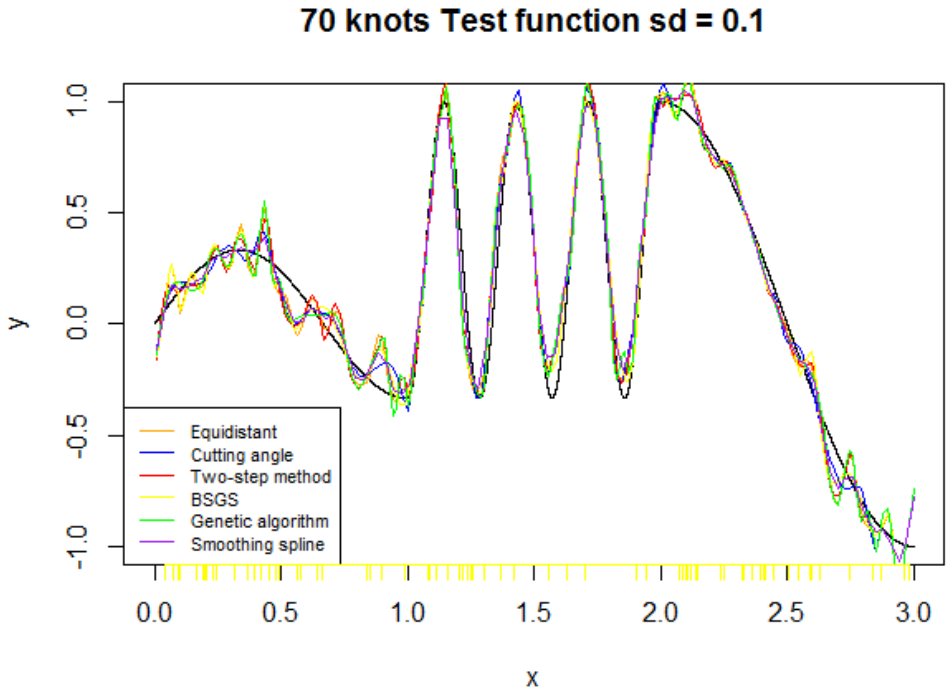
B.1.4 Visual results



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 17: Comparing the methods on Test function data samples ($\sigma = 0.1$).

B.2 Noise standard deviation $\sigma = \sqrt{0.001}$.

B.2.1 MSE

Table 21 The MSE of the different methods on data samples from the Test function ($\sigma = \sqrt{0.001}$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0756	0.0001	0.0020	0.0002	0.0017	0.0002
Cutting angle	0.0834	0.0118	0.0247	0.0192	0.0037	0.0213
Two-step method	0.0740	0.0044	0.0032	0.0025	0.0017	0.0002
BSGS	0.0801	0.0086	0.0233	0.0188	0.0022	0.0015
Genetic algorithm	0.0464	0.0029	0.0016	0.0003	0.0017	0.0002
Smoothing spline	mean		sd			
	0.0016		0.0003			

B.2.2 RSS

Table 22 The RSS of the different methods on data samples from the Test function ($\sigma = \sqrt{0.001}$).

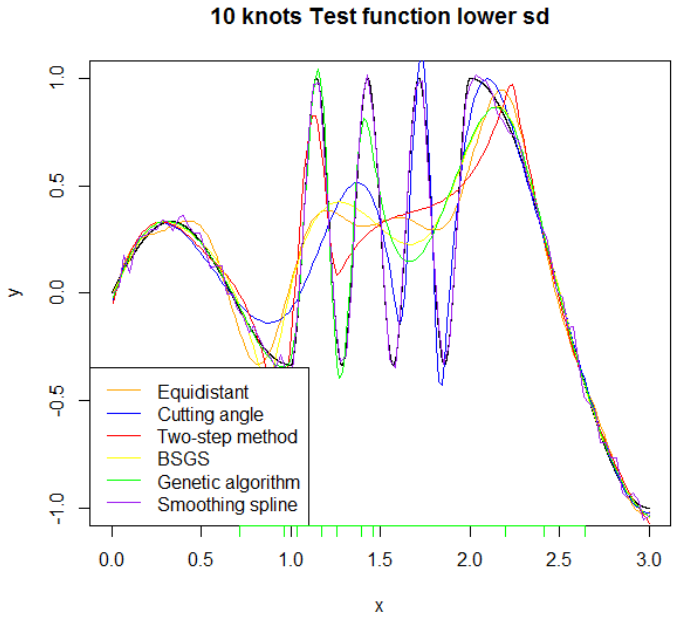
Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0760	0.0017	0.0014	0.0002	0.0002	0.0001
Cutting angle	0.0839	0.0120	0.0243	0.0195	0.0025	0.0222
Two-step method	0.0744	0.0050	0.0026	0.0026	0.0003	0.0001
BSGS	0.0806	0.0087	0.0228	0.0189	0.0008	0.0014
Genetic algorithm	0.0464	0.0030	0.0008	0.0003	0.0003	0.0001
Smoothing spline	mean		sd			
	0.0002		0.0002			

B.2.3 Computation time

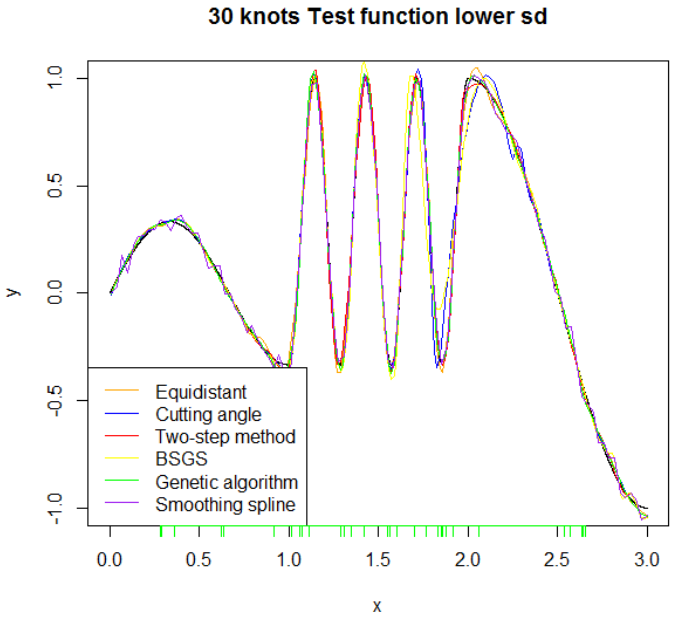
Table 23 The computation time (in seconds) of the different methods on data samples from the Test function ($\sigma = \sqrt{0.001}$).

Method	10 knots		30 knots		70 knots	
	mean	sd	mean	sd	mean	sd
Equidistant	0.0001	0.0010	0.0004	0.0027	0.0006	0.0033
Cutting angle	4.734	0.1049	5.644	0.5462	6.853	0.4441
Two-step method	2.388	0.0661	16.232	1.928	91.363	2.929
BSGS	0.7667	0.0642	1.293	0.0891	2.786	0.1372
Genetic algorithm	17.326	0.8007	31.098	1.105	70.545	1.759
Smoothing spline	mean		sd			
	0.0013		0.0045			

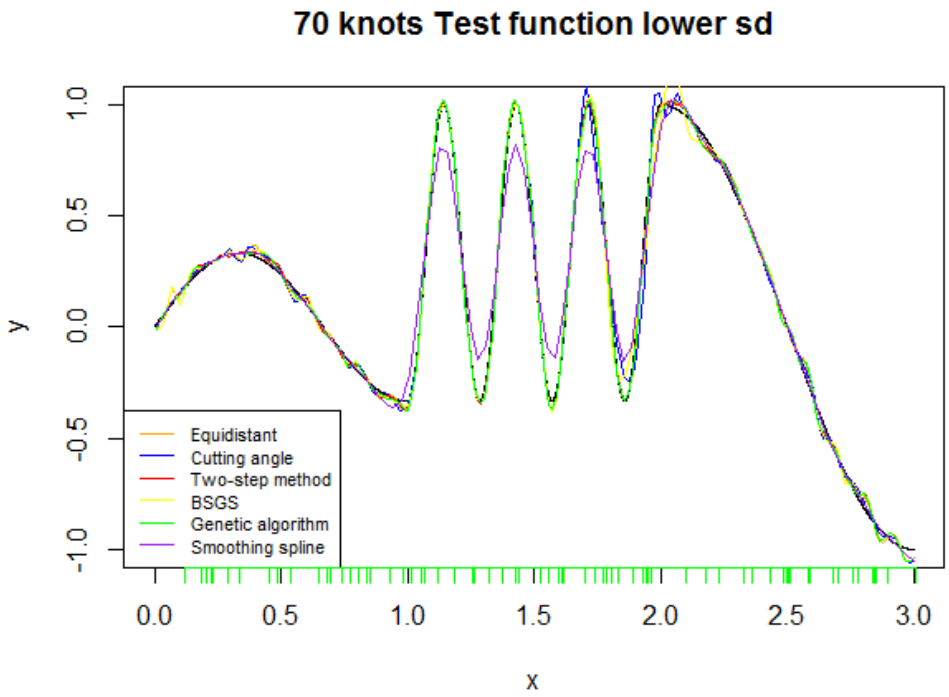
B.2.4 Visual results



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 18: Comparing the methods on Test function data samples $\sigma = \sqrt{0.001}$.

C Titanium

C.1 RSS

Table 24 The RSS of the different methods on the Titanium Heat data set.

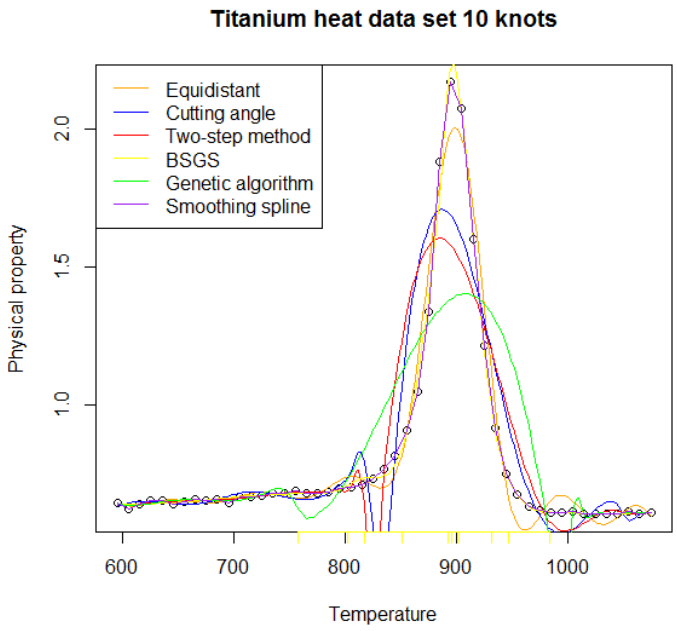
Method	10 knots	20 knots	35 knots
Equidistant	0.0037	$2.830 \cdot 10^{-4}$	$2.652 \cdot 10^{-5}$
Cutting angle	0.0240	0.0094	0.0002
Two-step method	0.0333	0.0093	0.0032
BSGS	0.0005	0.0026	0.0001
Genetic algorithm	0.0523	0.0047	0.0047
Smoothing spline	$1.243 \cdot 10^{-25}$		

C.2 Computation time

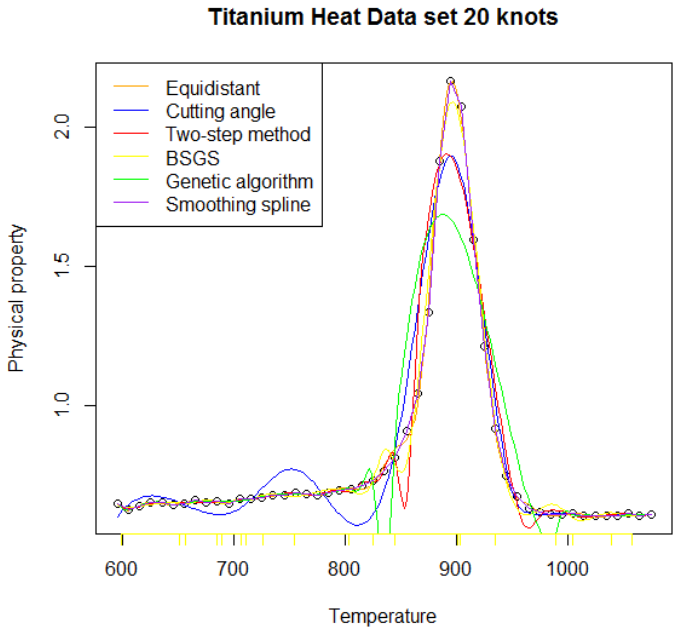
Table 25 The computation time (in seconds) of the different methods on the Titanium Heat data set.

Method	10 knots	20 knots	35 knots
Equidistant	0	0	0.02
Cutting angle	5.63	5.46	5.89
Two-step method	2.66	7.63	19.79
BSGS	0.64	0.83	1.16
Genetic algorithm	15.07	20.44	29.22
Smoothing spline	0		

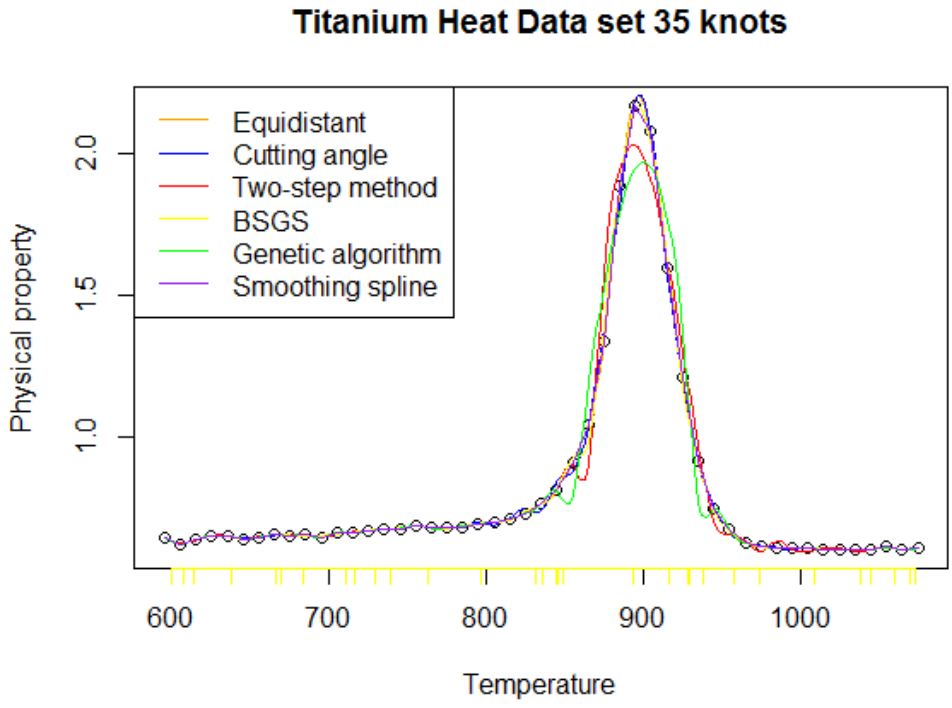
C.3 Visual results



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 20 knots.



(c) The different methods compared to each other for 35 knots.

Figure 19: Comparing the methods for the Titanium Heat data set.

D Simulated Motorcycle Accident

D.1 RSS

Table 26 The RSS of the different methods on the Simulated Motorcycle Accident data set.

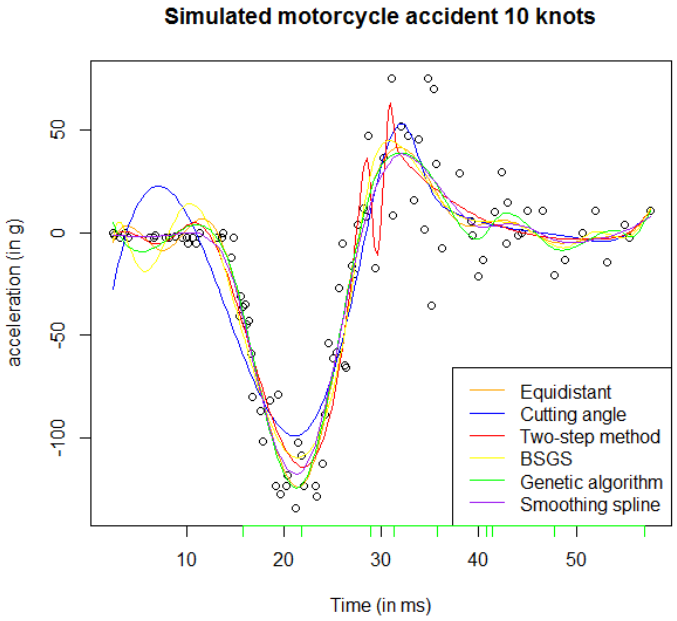
Method	10 knots	30 knots	70 knots
Equidistant	313.72	271.85	177.57
Cutting angle	523.09	270.64	192.58
Two-step method	345.99	293.02	171.19
BSGS	382.98	261.83	99.28
Genetic algorithm	308.52	197.92	36.43
Smoothing spline	313.64		

D.2 Computation time

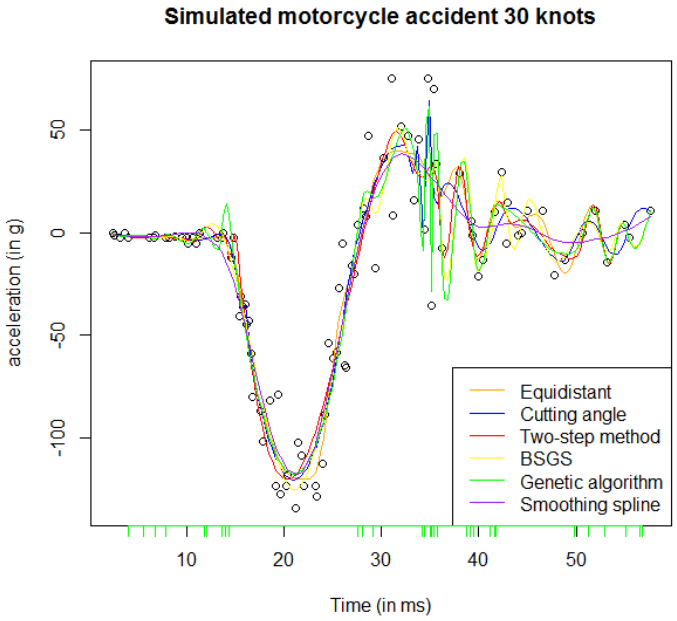
Table 27 The computation time (in seconds) of the different methods on the Simulated Motorcycle Accident data set.

Method	10 knots	30 knots	70 knots
Equidistant	0.01	0	0
Cutting angle	6.34	6.78	7.42
Two-step method	3.67	17.7	96.3
BSGS	0.75	1.25	2.61
Genetic algorithm	17.25	30.78	68.57
Smoothing spline	0		

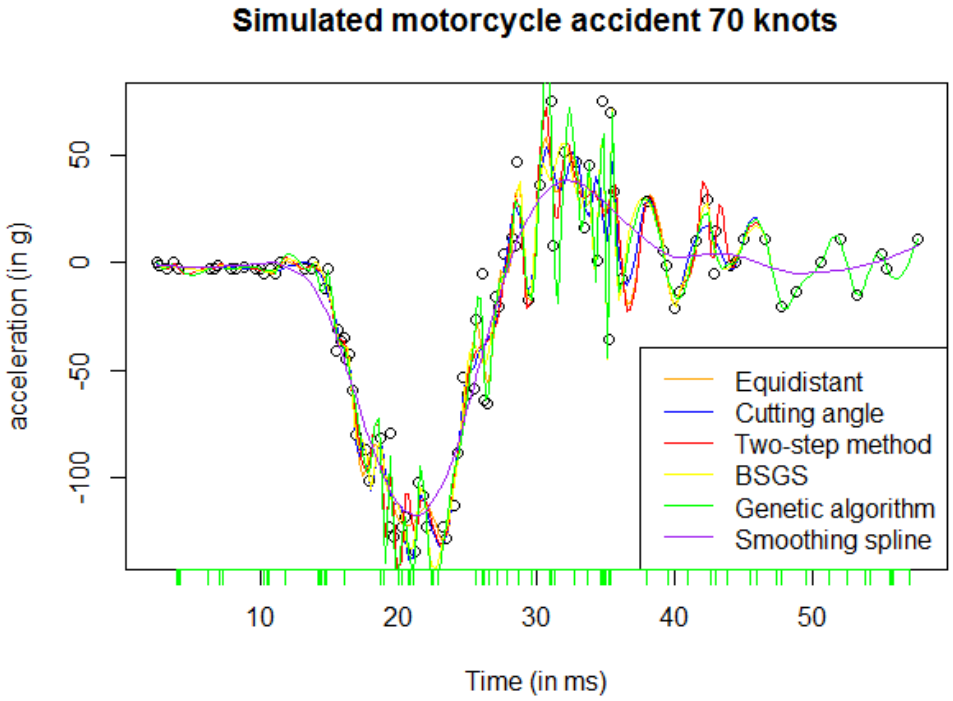
D.3 Visual results



(a) The different methods compared to each other for 10 knots.



(b) The different methods compared to each other for 30 knots.



(c) The different methods compared to each other for 70 knots.

Figure 20: Comparing the methods on the Simulated Motorcycle Accident data set.