DISSERTATION

Mathematical Methods for the Approximation of Radar Traces

Author:
Thilo SCHNEIDER, M.Sc.

Supervisor:
Prof. Dr. Tomas SAUER

Second Referee:
Prof. Tom LYCHE, Ph.D.

Submitted to the Faculty of Computer Science and Mathematics of the University of Passau

February 2015

Acknowledgements

This thesis would not have been possible if it had not been for the ongoing support of many persons. First of all I am greatly indebt to my supervisor Tomas Sauer. Our intensive and fruitful discussions, his ideas and feedback are the foundation this work is build upon. Whenever needed, he helped me to overcome both the scientific and administrative obstacles in my way and ensured a smooth progress of the research.

My gratitude is extended to Stefan Kunis. After my talk at the Rhein-Ruhr-Workshop 2014 he provided me with a reference to Björck [7] that in retrospective had a great influence on the results of this thesis. Sometimes even the little things have a big impact.

I am also grateful to Malcolm Sabin. His valuable input and ideas after the conference on New Trends in Applied Geometry 2014 made me think to include infrastructure related bounds and therefore gave rise to chapter 7.2.

This dissertation project was inspired and supported by the Fraport AG. I greatly appreciate that I was offered the opportunity to work on such a both mathematically interesting and practically useful problem. The option to spend parts of my working hours on research was invaluable for the success of this project.

In particular, I am deeply thankful to Steffen Wendeberg who is responsible for making all this possible. His pragmatic and supporting approach on managing my working hours ensured that I did finish the project on time.

Furthermore I would like to show my appreciation to my colleagues. Especially Andreas Figur certainly got some of the work assignments that usually would have been in my responsibility. By constantly covering my back he allowed me to concentrate properly on my research.

I had the honour to instruct several interns throughout the past years. I am thankful for the work of Christian Laus, Robert Hennings and Veronika Kilzer who implemented prototypes of my results, played with the parameterization of the algorithms and forced me to explain my knowledge in a way that frequently improved my own understanding.

My sincere thanks also go to the Erich-Becker-Stiftung that offered generous financial support for my research.

Finally my deepest thankfulness goes to my wife and my parents. Their support, encouragement and love is what kept me going in the long and dry periods of unsuccessful search for correct proofs. Especially my wife Sabine did not only have to remove my numerous spelling mistakes out of this work but also had to cope with a husband only talking mathematics during dinner.

Contents

Introduction

In November 2011, Frankfurt airport, one of the world's most important airports, opened its fourth runway. Part of the agreement between the airport's operator Fraport, the responsible regulatory authority and the surrounding population was that in exchange to the increased capacity caused by the fourth runway and the accompanying increase of noise pollution certain operational restrictions became effective. The most prominent of them was the introduction of a near total ban on night flights. Since then all aircrafts planned to depart on a certain day have to do so strictly before eleven o'clock in the night.

Within the first few months processes were not so well-rehearsed as they are now. Thus it frequently happened that aircrafts had to return back to their parking position after preparing for takeoff and taxiing to the designated runway.

But why did that happen? What could have been done differently by airport operations to minimize the risk of an aircraft being required to stay on the ground for the entire night?

Of course, the answers to these questions are neither straightforward nor to be searched at only one place, but rather are a composition of different conditions. One way to obtain the required information would be to regularly interview apron controllers and other involved employees. However, answers obtained this way are highly subjective and prone to interpretation errors.

This thesis lays out the foundations for an entirely different approach.

Most major airports collect positional information of the moving aircrafts via various technical means. Those positions then are brought to display to apron controllers and, in more advanced settings, used for alerting and guidance purposes. While those *Advanced Surface Movement Guidance and Control Systems (A-SMGCS)* [2] primarily are used for real time purposes, the collected data of course could be saved and used later to analyze situations in the past.

As we may see in the example above there might be a wide range of questions that could be attempted to be answered by such a collection of data. Those include, but are not limited to:

- How is the traffic on the airport distributed?
- What is the velocity aircrafts are travelling with at specific points?
- Where do delays occur and how are they caused?
- What are typical areas where aircrafts are held on during their taxi process?
- How do aircrafts relate to each other: What distances do they keep and how does one movement depend on another one?

Figure A: Frankfurt's A-SMGCS constantly visualizes the position of moving aircrafts to the responsible apron controllers. (Photo: Fraport AG)

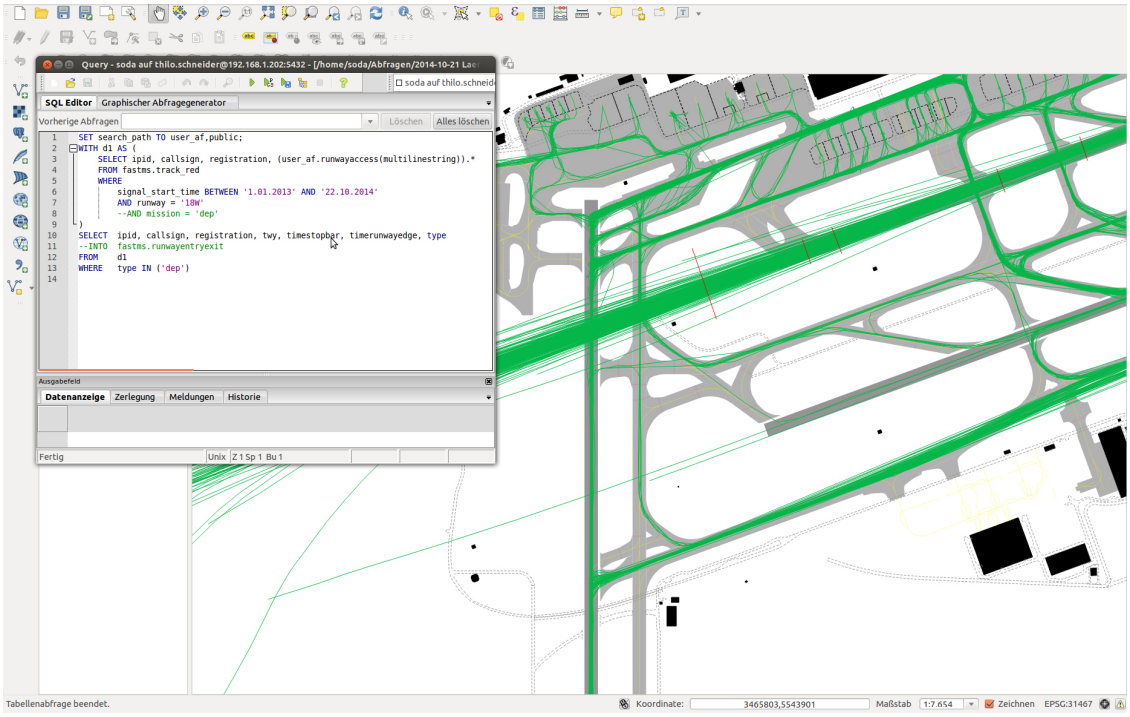


Figure B: Screenshot of Fraport's analysis toolkit for A-SMGCS data that is build upon various open source software products.

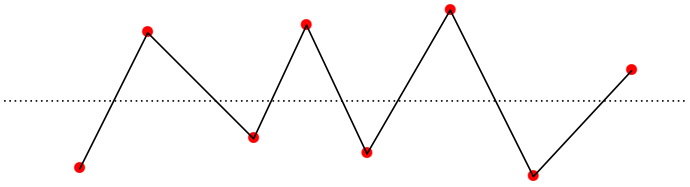


Figure C: If one just linearly interpolates the recorded positions, it is immediately clear that velocities between data points can be greatly overestimated.

For most of those questions it is favourably to analyze a significant amount of distinct movements at once to gain statistical significance. Preferred are assessments that take a time span of a year or at least half a year into account.

Fraport AG developed an in-house solution [31] to analyze the available A-SMGCS data. This software toolkit is flexible enough to model most of the arising questions and is able to query even large amounts of data within reasonable running times.

Due to technical limitations of the A-SMGCS the obtained positional accuracy is not as good as one could wish. While those inaccuracies usually do not pose a problem in real-time use of the data, they do in post-processing. One of the arising problems is demonstrated in Figure C, where the velocities are overestimated if the data points are just interpolated. Much better results could be gained if some kind of approximation was used that smoothes the positions. In summary: For attaining the possibility of a nearly automatic data processing it is important to deal with data errors sufficiently.

Before this thesis the smoothing algorithm employed by Fraport AG was a standard smoothing spline implementation without much further modelling. Whereas this approximation delivered reasonable results for the majority of the data, it turned out that it performed poorly in certain situations.

This thesis strives to develop mathematical methods improving approximation quality up to a level that fixes data problems in a way they do not interfere with later analysis. From a mathematical point of view the central challenges to be overcome are the use of robust smoothing methods and modelling physically correct behaviour around the standstill of aircrafts.

The following research will primarily focus on the *ground movement* of aircrafts. Nevertheless, the developed methods should be applicable to positional information of all types of vehicles that follow similar physical laws, especially cars, trucks and busses as well as ships to a certain amount.

The content of this thesis is divided into three parts. The first part seeks building a model of the available data. The second part introduces the mathematical techniques used on an abstract level. Finally those two components are combined in the third part by formally explaining the steps required for data smoothing.

Chapter 1 strives to give an in-depth introduction to the problem and the available data. As such it is the only chapter that has a strong connection to airport processes and airport know-how. We will focus on understanding the data, its characteristic problems and the corresponding challenges. At the end of the chapter a more formal model will be built.

In direct succession chapter 2 introduces the actual algorithm we will use on a heuristic and descriptive level. The main goal is to give the reader the knowledge necessary to answer the question *"Why is this specific mathematical theory necessary for the application?"* throughout the remaining parts of the thesis. Up to this point the text is intended to be understandable to any reader with high school level mathematical background.

We will use *B-splines* as the central concept to represent aircraft movements. Chapter 3 briefly introduces the relevant theory and states central results that are available in literature. The reader familiar with the topic may skip this chapter altogether.

Chapter 4 establishes various approximation concepts on an abstract level. While those techniques will be used in the context of spline approximation, the assumptions required throughout this chapter are kept as general as possible. From a scientific point of view one of the major contributions of this thesis to science can be found within chapter 4.2: Here a rigorous treatment of the convergence of robust approximation methods is conducted. Although parts of this analysis have been known for multiple decades the work presented in the following seems to be the most exhaustive up to the current knowledge of the author.

It turned out that proper treatment of phases the vehicle under observation is standing still is of central relevance for the quality of the approximation. The last part of this thesis is opened by the discussion on how to detect those standstill positions in chapter 5.

The application of the approximation concepts to spline smoothing is introduced in chapter 6. Again, this chapter serves as a repetition of known results and may be skipped by the familiar reader.

Another major question of this thesis is how to handle standstill periods with various geometric restrictions in the spline smoothing process. Those methods are discussed in chapter 7.

Chapter 8 concentrates on how to approximate the segments between standstills. Among others it is discussed how to obtain optimal knot sequences for the spline to be saved. This method is strongly based on observations by de Boor and generalizes his work in two small, but relevant aspects.

Finally, chapter 9 summarizes the results and shows examples of the smoothing results obtained. Here also the issue of the performance of the algorithms developed is broached. At last the author offers ideas for further research.

Even if mostly common notation is used, it is not avoidable to define central symbols used in multiple occasions. References for those symbols and notation are listed in the appendix.

Part I

Preliminaries

1 Defining the challenge

Before we may attempt to develop algorithms dealing with the available data, it is advantageous to gain some basic understanding of the data itself, including its characteristics, drawbacks and specialities. We will spend the next paragraphs to give a brief introduction to the technology employed. While this is not strictly necessary for the understanding of the remaining parts of this thesis, it provides useful background information that help the reader to integrate the developed algorithms into a greater context.

1.1 The data supply system

The positional data collected by Advanced Surface Movement Guidance and Control Systems usually is a fusion of multiple data sources that originate from up to four different technologies:

- Positions from a *Multilateration System (MLAT)*. Multilateration is based on the idea that any vehicle equipped with a suitable transponder regularly transmits a signal with various status information of the vehicle. Those signals are collected by multiple receivers spread over the observed area and the vehicles position is calculated based on delay differences. As long as the signals are not disturbed this system probably offers the best accuracy. More information can be found in the book *Multilateration - Executive Reference Guide*, available online [27].
- *Radar signals*. Here, the images from different radar systems are processed and the centre of masses of the aircraft point clouds are computed. This approach offers very high detection rates and good stability, whereas the positional accuracy generally is limited to the accuracy of the centre of masses. This in turn highly depends on the angle the vehicle is rotated to the radar as well as weather conditions and other factors influencing the quality of radar images. At most airports there are different radar systems in use that cover both ground and surrounding airspace.
- *Automatic Dependent Surveillance – Broadcast (ADS-B)* data. This is a position information most modern aircrafts broadcast regularly and that can be received by anyone with suitable hardware. The quality of the signal heavily depends on the accuracy of the position the aircraft's navigational systems provide. Depending on the available equipment the information originate either from the aircraft's GPS device or from dead reckoning. In this case the position constantly is updated based on the known velocity and movement direction. GPS coordinates generally are mostly accurate, but there is no such guarantee for the data from dead reckoning. As ADS-B data can easily be received there exist free web portals visualizing all positions of aircrafts in certain parts of the world [30].

- *Gap-filler technologies*. Due to the limitations of radar and ADS-B data on the ground the data from the MLAT system is of special importance. However, upgrading the system to increase coverage in a narrow area is cost intensive and thus tried to avoid. One possibility to do this is by employing technologies that allow the localization of vehicles through other means. Prim candidates are surveillance camera systems and induction loops. As those gap fillers are not yet used at Frankfurt Airport their effects are neglected in this thesis.

The A-SMGCS collects data from some or all of the data sources mentioned above and tries to create a fused position of those information. The algorithms employed are proprietary and not known to the author of this thesis. We will treat the fusion process mostly as a black box. Due to observations we may assume that the fusion performs reasonably well when all data sources are present but shows some quirks when signals are lost and extrapolation is used to compute positions.

This thesis primarily intends to develop algorithms that are able to deal with A-SMGCS data. However, the developed algorithms should also be able to process MLAT or radar data directly. Even if all systems have their specific weaknesses it turns out this generalization does not harm the algorithms performance in any way.

In the following we will use the term *radar data* for all those data sources interchangeably. Although this denomination is slightly imprecise it probably is the most natural term for readers with non-aviation background. This simplification also was done in the title of this thesis.

1.2 The data itself

Depending on the data source various data attributes are available. The most important and generally available fields are the x- and y-coordinates in a defined spatial reference system, the time of measurement and, depending on the vehicle under observation, the height. Furthermore some systems provide an estimated velocity that originates from the aircrafts inertial navigation system. The resolution of those values is limited by the system specification. The system employed in Frankfurt, for example, generates the position value with an accuracy of 0.25 meters.

The remaining attributes consist of various operational and technical data, such as an unique address of the aircrafts transponder, attributes to identify the aircraft, possibly the type of the aircraft and so on. While this information is vital for analyzing the data, it is of no relevance to the problems in this thesis.

The frequency the positional information are recorded again depends on the data source. In Frankfurt the position of a vehicle is determined approximately each second by the A-SMGCS system. In the surrounding airspace we have to rely on radar data which, restricted by the rotation velocity of the radar, comes only every five seconds. It is always possible that single recordings are left out or are duplicated.

Figure 1.2 shows some typical movement patterns that can be encountered frequently. Different phases of the aircraft's movement process usually evoke different phenomena to be noted within the data. This is mostly related to the fact that the technical systems collecting the data behave differently in certain phases, e.g. the position of an aircraft in flight is detected mainly by radar while on the ground the main source is multilateration. While the transition between those states

time	x-coordinate	y-coordinate	height
956	-4164	-1184.75	152.4
957	-4095	-1159	152.4
958	-4041	-1140	152.4
959	-3993.25	-1124.5	144.78
960	-3926.75	-1100	144.78
961	-3860	-1075.25	144.78
962	-3793.5	-1050.75	144.78
963	-3744.25	-1033.25	144.78
964	-3693	-1018.75	129.54
965	-3628.5	-995.5	129.54
966	-3582.75	-977.25	121.92
967	-3521.75	-953.75	114.3
968	-3465	-932.5	114.3
969	-3400.25	-908.75	106.68
970	-3336	-886.75	99.06
971	-3269.75	-866.25	99.06

Table 1.1: Subset of data from an arbitrary aircraft movement. The time coordinate is measured in seconds from the beginning of the recording; the spatial reference system used for the x- and y-coordinates is local to the aerodrome reference point[1].

is floating, it is helpful having a separate look at the typical phenomena that can be observed. The following information is based on intensive empirical studies of the data but does by no means claim to be complete.

In flight, during takeoff and landing

The target is moving with a high velocity. Compared to this velocity the observable noise of the data points is very small. A smooth movement is observable in most of the cases.

Jumps in the data happen frequently, most of the times at transition points where different data sources start or end to account to the reported position. This usually includes some correction in the reported positions.

It is difficult to verify the absolute data accuracy. According to the observations made it is likely that there are systematic and non-random errors. Due to the nature of approximation we will not be able to deal with those problems though.

Taxi with medium to high velocity

A smooth movement is clearly distinguishable, the data points seem to be scattered along a smooth curve. The observable errors appear to be randomly distributed and independent of each other. A straightforward approximation of those data points promises to achieve good results.

[1] The *aerodrome reference point* is the origin of the airports local coordinate system. It is fundamental to all aspects of airport design and one of the first things to define when planning a new airport [54].

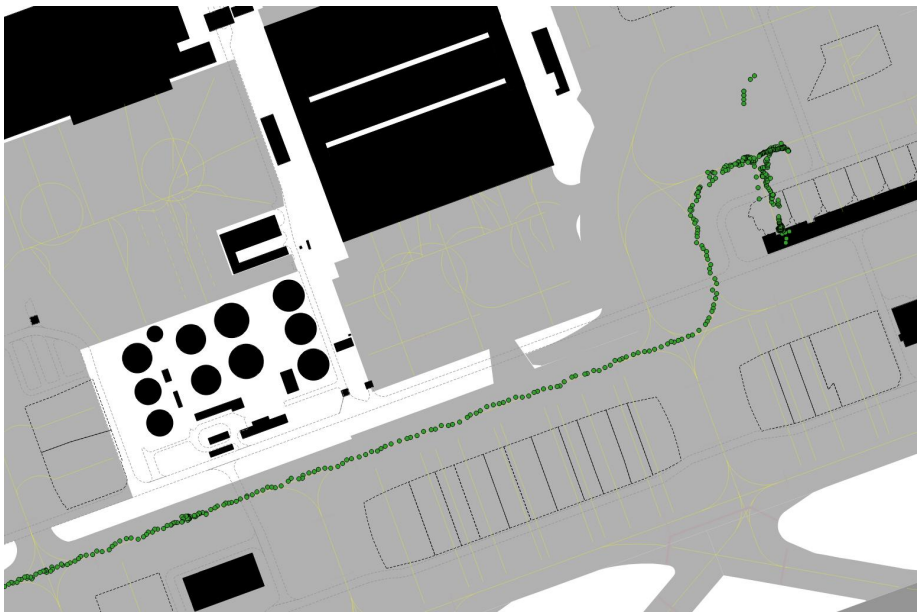


Figure 1.1: A typical aircraft movement with the airports infrastructure shown in the background. As the infrastructure is of no relevance to the algorithms proposed we will mostly omit this kind of display and use standard scatterplots.

Greater data problems do occur rarely. The most common of those rare problems are gaps in the data. Short gaps, e.g. one to three missing data points usually do not influence the position accuracy of surrounding points. In case of longer gaps it is likely to have up to five massive outliers in the direct neighbourhood of the observable gap.

Pushback[2], taxi with low velocity

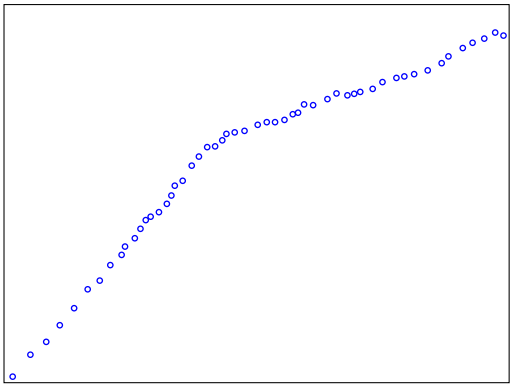
Compared to movements with higher velocity the variance of the random errors around the assumed smooth curve greatly increases. This pattern is often observable together with gaps. Massive outliers are common and may happen frequently. Most of the time outliers appear in groups of two to four in a row. The increase of errors partially is based on the fact that due to the proximity of buildings and other vehicles a lot of signal reflections appear.

Parking or holding

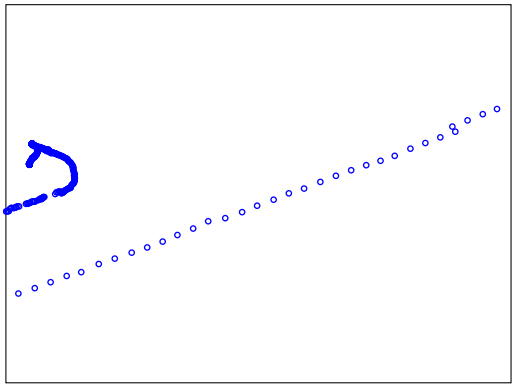
The variance of errors comes to a peak when the target is not moving anymore. Frequently the measured velocity between single data points equals the velocity observed at aircrafts during flight. Apart from that, gaps and massive outliers may be observed regularly. Again outliers emerge in groups of up to five consecutive points. During standstills however it happens that two or more such outlier groups directly follow on each other.

At times the centre of the data is oscillating between two different positions where one can be clearly identified as being the correct location as the data points prior and after the holding position lead to this position.

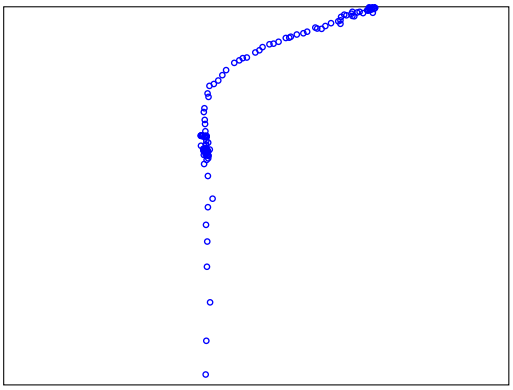
[2]The process where a departing aircraft is pushed back from its parking position by a separate pushback tractor is called *pushback*. The pushback process continues until the tractor has been decoupled from the aircraft.



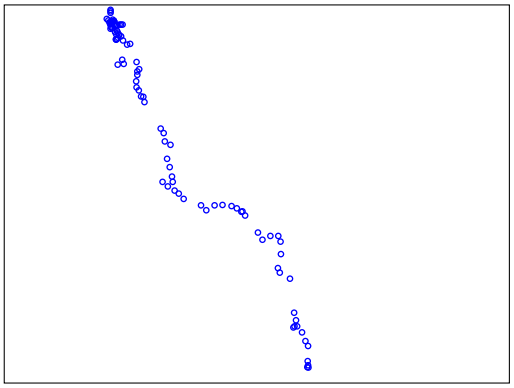
(a) Typical, well-behaved aircraft movement on the ground.



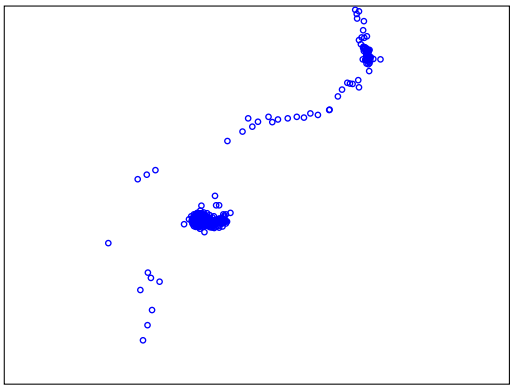
(b) Aircraft in final approach. The jump where a new data source becomes available can be clearly noticed on the right.



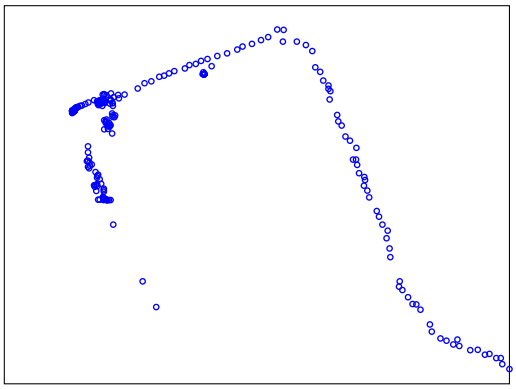
(c) Aircraft at takeoff. Due to the velocity change the spacing of data points increases significantly.



(d) More scattering, the standstill position is good to recognize.



(e) Standstill with a gap connecting it to the rest of the data points, heavy outliers around the holding point.



(f) Movement with two standstills, one point is oscillating between different positions.

Figure 1.2: Examples of characteristic movement patterns. Each figure shows the projection on the x-y-plane and thus neglects the time component of the data points.

If the halt is at the start or end of the movement it is typical that the first or respectively last few points are outliers.

While airport operations carefully distinguish between parking and holding the results regarding our data are identical. Therefore, in the future we will refer to those processes in summary as *standstill* or *standstill period*.

1.3 Design goals and model building

The algorithms that shall be developed should be able to correct the data problems mentioned above as good as possible. Apart from that it is crucial that the algorithms are able to deal suitably with a wide range of situations. In particular, some care is needed to avoid the algorithms to fit to the given problems so closely that generalization is difficult. Nevertheless the list above can be reduced to a few different kinds of problems the algorithms will have to be calibrated for:

- **Detection and handling of standstills:** To enforce the fitted function to be absolutely constant it is necessary to detect standstill periods and take care of them in an appropriate manner. At the same time this detection allows to take out the parts of the movement that probably have the worst data quality and therefore impede further processing.
- **Handling of non-uniformly distributed parameter values:** As there are gaps and duplicated points in the time-domain of the data the algorithms will not be allowed to assume a strictly uniform distribution of the data points in the parameter space.
- **Removal of outliers:** Those points influence all algorithms that assume a common distribution and independence of errors. Thus we will need to present a method to detect those outliers and either remove them or incorporate them in the following steps in a way that they do have not to much influence on the final results.
- **Handling of random noise:** The core concept of the approximation is to smooth out the noise mentioned before while not loosing details in the data that come from actual movements of the vehicle.
- **Varying significance of data changes:** On the ground and with slow motion an error of ten meters is significant in a way that it may alter following analyses. In contrast the same error in the airspace and at much higher velocity is barely noticeable. Hence all parameters that influence the approximation's accuracy should be designed in a way that allows changing them throughout the movement.

Having carefully defined what we would like the developed algorithms to achieve and how this corresponds to the data at hand we are now able to deploy a suitable notation as well as a model used for analyzing the data.

Our general goal is to find a function f mapping the time to the position of the aircraft.

We will rely on the following notation:

- The vehicle movement is denoted by $f\colon \mathbb{R} \to \mathbb{R}^l$, $l \in \{2, 3\}$, where the first component of the functions value denotes the position on the x-axis and the second component the position on the y-axis of the coordinate system. If the height coordinate is present, the third coordinate will be used to give the representation for the height. Depending on the context we will use f either for the true and unknown vehicle movement or the approximation.
- The single components of f will be written as f_x and f_y and, if available, f_z, that is $f = [f_x, f_y]^\mathsf{T}$ or $f = [f_x, f_y, f_z]^\mathsf{T}$. Here the symbols x, y and z are just a symbolic reference to the respective axis of the coordinate system. Whenever used in subscripts, we will use this interpretation.
- The data points have been recorded at the sites $t_0, \ldots, t_{m-1}$ resulting in the known positions $P_0, \ldots, P_{m-1} \in \mathbb{R}^l$. If we need the single elements of P_i, we will assume $P_i = [P_{x,i}, P_{y,i}]^\mathsf{T}$ or $P_i = [P_{x,i}, P_{y,i}, P_{z,i}]^\mathsf{T}$.
- In vector notation those are written as $\mathbf{T} := [t_0, \ldots, t_{m-1}]$ and $\mathbf{P} = [P_0, \ldots, P_{m-1}]^\mathsf{T}$. Whenever required we additionally will interpret $\mathbf{P}$ and $\mathbf{T}$ as sets containing the values $t_0, \ldots, t_{m-1}$ and $P_0, \ldots, P_{m-1}$ respectively. There will be no confusion on the version used in a specific situation. Furthermore, we define the dimension-wise vectors $\mathbf{P}_x := [P_{x,0}, \ldots, P_{x,m-1}]^\mathsf{T}$, $\mathbf{P}_y := [P_{y,0}, \ldots, P_{y,m-1}]^\mathsf{T}$ and $\mathbf{P}_z := [P_{z,0}, \ldots, P_{z,m-1}]^\mathsf{T}$.

Using the introduced notation we may describe our regression problem as

$$P_i = f(t_i) + e_i \tag{1.1}$$

where e_i represent the random, possibly correlated noise. We will assume the expected value of e_i to be zero.

The main work in this thesis will be dedicated to finding an estimate for f. Criteria that can be used to judge the performance of the algorithms are as follows:

- **Correctness in the absence of noise and outliers.** Is the result of the algorithm correct if $e_i = 0$ for $i = 1, \ldots, m$?
- **Correctness in the presence of random noise but without outliers.** Does the algorithm perform well in case that the errors e_i follow a symmetric, uncorrelated and reasonably well located distribution?
- **Robustness against outliers.** What happens when outliers are present?
- **False positives.** Whenever something is detected, how are the chances that we detect something we did not want to identify?
- **Performance.** How fast can the computations be done? What is computational complexity of the algorithm?
- **Storage requirements of the approximated function.** How much memory does the approximated function require to be stored?

We will either design our algorithms with those model assumptions in mind or come back to those criteria and to some degree will verify the developed algorithms against those requirements.

2 Introducing the algorithm

In chapter 1.2 we observed that the quality of the radar data directly corresponds to the vehicles movement state. It proves useful to distinguish between those phases within approximation mainly for this reason. Especially standstill periods have to be identified and treated separately as they not only require specific algorithms but also offer great potential for performance optimization of the algorithm.

Phase detection

Detection whether an aircraft is in flight or on the ground does not require mathematical algorithms, because the transition between those phases is bound to happen on runways. Thus, it is easy to filter out all data points located on the relevant runway and define the corresponding time interval as *takeoff* or *touchdown*. For our purpose it is sufficient to identify this transition phase and not to detect actual takeoff or landing times.

Things get more difficult when we try to detect standstill periods. The infrastructure-based approach we took for takeoffs and touchdowns does not work sufficiently anymore, as it is possible that the vehicle stops moving at any place while on the ground. Furthermore, it will be necessary to add the fact that the vehicle is not moving as a boundary condition to the approximation problem later on for various reasons. Thus, we have to detect standstill periods as accurately as possible.

Therefore we will use a two-step approach: Firstly we identify *standstill candidates* that are guaranteed to include the standstill period and then we will use various methods to refine the exact standstill location until we yield a good estimate on the correct standstill period.

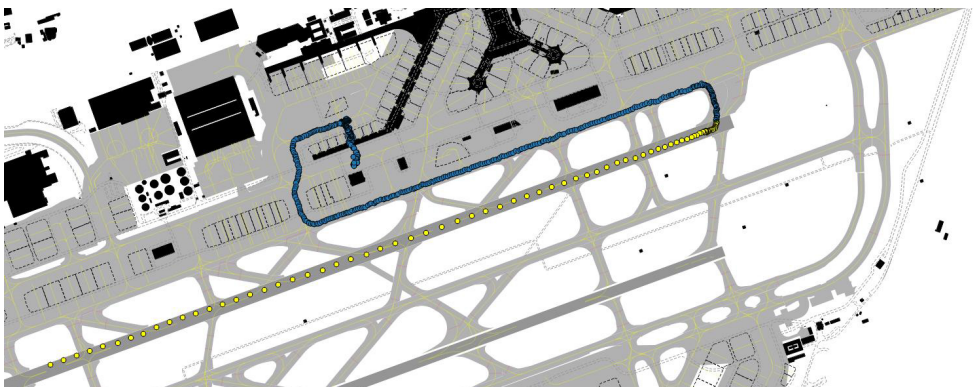


Figure 2.1: The transition between flying and taxiing on the ground may be located by identifying the points geometrically located on a runway.

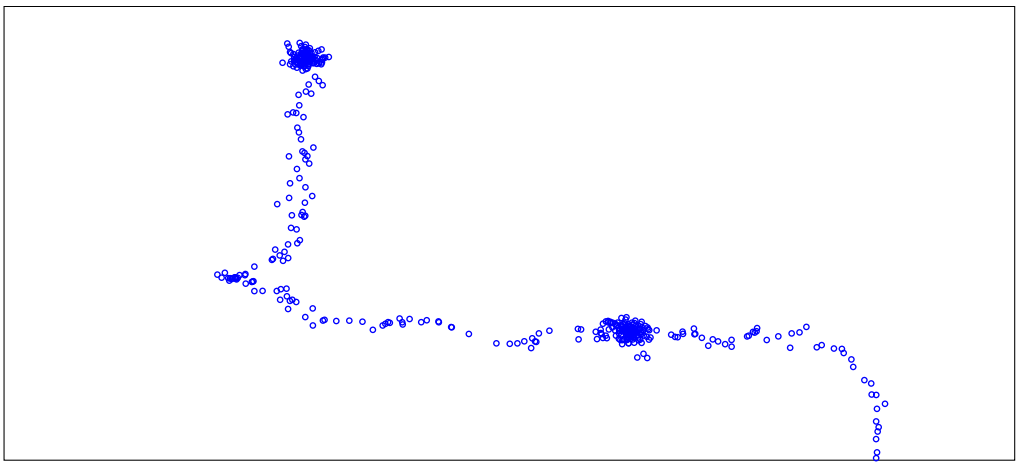


Figure 2.2: The human mind is able to analyze the pattern in the data instantly and provides us with a good guess of the standstill's location.

A standstill of relevant length is characterized by the fact that there are multiple data points at various times and roughly the same location. Thus, an intuitive way to detect whether data points might belong to a standstill is to count how many other points lie close to it. If this number exceeds a given threshold the point is likely to belong to a standstill. We will formalize this idea in chapter 5.

The obtained standstill candidate will need to be refined in the later processing. Initially we use a straight linear regression line to narrow down the approximated beginning and ending of the standstill. Afterwards we will fine-tune the standstill duration by repeatedly approximating it with different standstill lengths and choosing the ones which result in the best fit in a close neighbourhood of the standstill location.

Approximation of standstills

From a mathematical point of view the correct approximation of standstills forms the major challenge of this thesis. Describing what has to be done, on the other hand, could not be more straightforward.

We distinguish between four types of standstills that each need separate care:

Standstill at the begin or end of movement. In this case we just have to force the approximation to be constant within the standstill interval. Comparing Figure 2.3 (a) to Figure 2.3 (b) shows that this is necessary to yield good results.

Standstill at the begin or end of movement with known direction. Data recordings commonly start or end at parking positions of vehicles. These parking positions provide further information in which direction the movement has to begin. This especially is helpful as those parking positions usually are close to buildings and therefore come with worse data quality than usual. However, due to infrastructure limitations, we can assume that the vehicle has been moving in a certain, known direction. Adding this condition to the

approximation process may increase the approximation quality further, as can be seen in Figure 2.3 (c).

Standstill between periods of movement. It is of course possible to approximate a standstill between periods of movement just by adding the boundary condition enforcing the approximation to be constant as was done above. This leads to an approximation shown in Figure 2.3 (d). To get a behaviour that is more likely to resemble the actual physical trajectory of the movement we might add another boundary condition that enforces the movement directions before and after the standstill to be equal. The result is visualized in Figure 2.3 (e).

Standstill as a way to model the pushback process. During the pushback process an aircraft is moved backward by an external force. Afterwards, it either is pulled forward again or the tractor is decoupled and the aircraft continues its track with its own engines. In any way the movement direction after the backward movement changes by exactly 180 degrees. It turns out that modelling this special case results in the same boundary conditions as in the case of standstills between periods of movement. Thus, we need not to take special care. However, we have to keep in mind that the holding time between moving backward and forward might be very short as the towing vehicle might only need to change directions.

Approximation between standstills

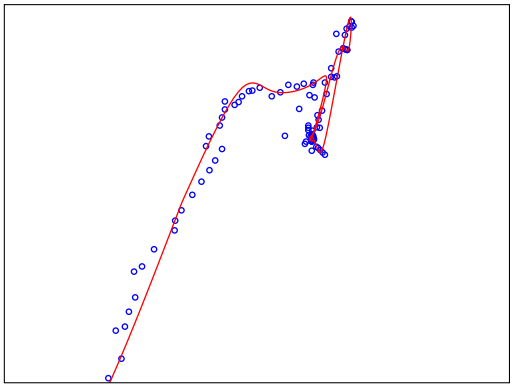
Of course we now could add the derived boundary conditions for all standstills to a combined approximation problem that fits the whole vehicle movement all at once. However, as we already approximated the proximity of detected standstills while searching for the exact standstill intervals we might just use this information.

In fact, it is sufficient to approximate the segments between standstills that have not yet been approximated and then combine these different parts to one big approximation spanning the whole vehicle movement. This way we not only save time by not approximating the standstills again but also reduce computational complexity in total.

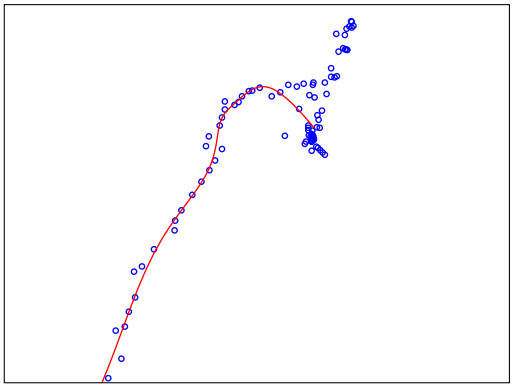
This is caused by the fact that within the approximation process at least one system of linear equations has to be solved, or, in other words, a matrix decomposition has to be computed. With most implementations this is an process that needs $O(m^3)$ operations with m being the number of observations. While in theory faster algorithms exist, they are hardly used for various reasons [60, p. 108]. Hence, if we use a detected standstill period to split up the approximation problem in two parts with m_1 and m_2 data points, respectively, the approximation process only needs $O(m_1^3 + m_2^3)$ operations. Depending on m this might be a significant gain in performance.

In case there are longer data gaps within the segment to approximate we have no information on how the vehicle behaved during this times. Furthermore we already know that those gaps are not within a standstill period. Thus, depending on the length of the gap it probably will not be possible to fill in the missing information. The best strategy available is to break the part up and keep the gap in the resulting approximation. For later analysis it is better to have no data than incorrect data.

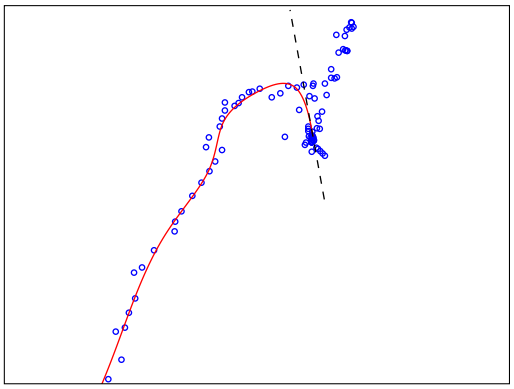
There is one final aspect to keep in mind. In chapter 1.3 we formulated the requirement that the approximated function can be stored with a minimal amount of required memory. Thus, to reduce



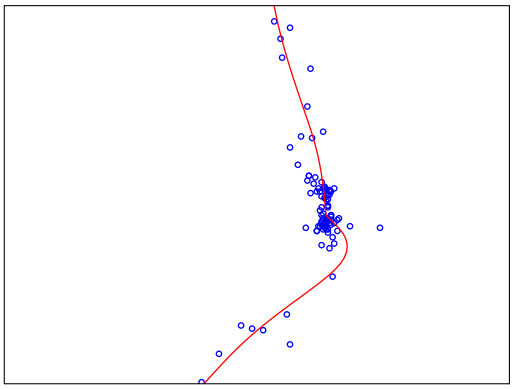
(a) Without special care the approximation will not be constant over the whole standstill interval.



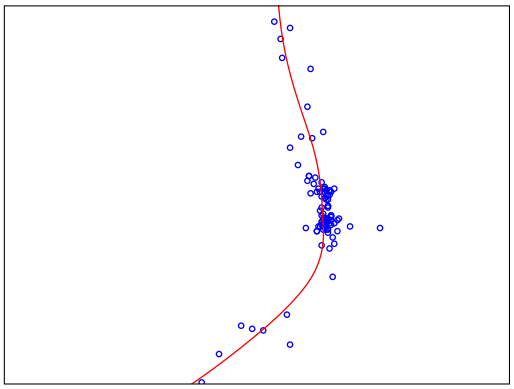
(b) Forcing the approximation to be constant within the standstill yields much better results.



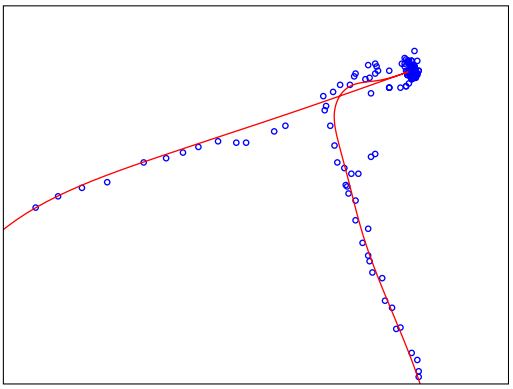
(c) Forcing the direction to equal the infrastructure direction may increase approximation quality further.



(d) If the standstill is between periods of movement, the movement direction may change after the standstill. The resulting curve is not differentiable.



(e) Forcing the directions to be equal results in an approximation that usually resembles the actual physical behaviour of the vehicle better.



(f) The same process allows a good approximation of the pushback procedure.

Figure 2.3: Various standstill situations have to be treated differently.

storage needs we furthermore will show how an already found approximation can be used to find a new approximation with fewer parameters to save.

Algorithm outline

Summarizing the discussion above we gain the following, rough algorithm outline:

1. Detect all standstill candidates and obtain rough bounds for the standstill.
2. For each found standstill candidate:
   (a) Approximate using the appropriate boundary conditions.
   (b) Optimize the standstill duration such that the region around the holding position is represented best.
3. For each segment between two standstills:
   (a) If there is a long gap without data points within this segment, split the segment up and do the following steps with each part of the segment separately.
   (b) Approximate with boundary conditions derived from the standstill approximations.
   (c) Approximate again with optimized parameter count.
4. Combine the single approximation parts to one big approximation.

In the following we will develop the single steps of the algorithm in depth and present the mathematical theory that assures correctness of the suggested methods.

Part II

Core concepts

3 Spline curves

Representing arbitrary types of data points by a curve requires a suitable mathematical description of such curves. While there are quite a few possible ways to define sufficient ones, we will use one special class, namely *spline curves*.

Splines stand out due to a few properties that make them remarkably useful for the problems at hand:

- They offer a compact representation using relatively few parameters that have a clear geometric interpretation.
- A strong mathematical structure that helps deriving required results is available.
- Differentiation of spline curves is easy and reduces the complexity of the function.
- Efficient algorithms for computing the spline's values as well as its derivatives exist.

Splines are considered first to be brought to attention by Schoenberg in 1946 [15], though even nowadays they are still a topic of intensive research. A few decades later, in 1978, Carl de Boor published a rigorous treatment of fundamental results which still is one of the most comprehensive books on that particular topic. Thus, the following chapter is manly based on the revised edition of his *Practical Guide to Splines* [10]. Additional information may also be found in Farin's book *Curves and Surfaces for CAGD* [29]. More recently, Klaus Höllig and Jörg Hörner published *Approximation and Modelling with B-Splines* [40], an excellent and understandable introduction to the topic.

One way of representing spline curves is using a linear combination of certain basis functions:

$$f(x) = \sum_{i=0}^{n-1} d_i N_{i,q}(x).$$

In this chapter we will first introduce the basis functions $N_{i,q}$ and examine some of their most important features. Afterwards a formal definition of spline curves becomes possible. As a last step, we will present a fundamental algorithm for the evaluation of spline curves.

3.1 B-splines

Definition 3.1. (a) For given $n, q \in \mathbb{N}$ the vector $\mathcal{T} = [\tau_0, \ldots, \tau_{n+q}]$ with $\tau_i \leq \tau_{i+1}$ for all $i = 0, \ldots, n+q-1$ is called a *knot sequence*.

(b) The elements of $\mathcal{T}$ are called *knots*.

(c) If $\tau_{i-1} < \tau_i = \cdots = \tau_{i+k-1} < \tau_{i+k}$, then τ_i is a knot of *multiplicity k*.

Now, using the terminology just introduced, we may define the basis functions that will form the building blocks of the spline curves.

Definition 3.2. For given $n \in \mathbb{N}$, a *knot sequence* $\mathcal{T}$ and $i \in \{0, \ldots, n-1\}$ the *i-th B-spline* of *degree q* is recursively defined as

$$N_{i,0}(t \mid \mathcal{T}) := \begin{cases} 1 & \text{if } \tau_i \leq t < \tau_{i+1}, \\ 0 & \textit{otherwise}, \end{cases}$$
$$N_{i,q}(t \mid \mathcal{T}) := \frac{t - \tau_i}{\tau_{i+q} - \tau_i} N_{i,q-1}(t \mid \mathcal{T}) + \frac{\tau_{i+q+1} - t}{\tau_{i+q+1} - \tau_{i+1}} N_{i+1,q-1}(t \mid \mathcal{T}). \tag{3.3}$$

The *order* of the B-spline equals $q + 1$.

We note that (3.3) may contain division by zero for suitable knot vectors. However, in that case the corresponding $N_{i,q-1}(t \mid \mathcal{T})$ or $N_{i+1,q-1}(t \mid \mathcal{T})$ will be 0 or undefined as well. Therefore it is adequate to define the whole product to equal 0 in this rare case.

Convention 3.4.
1. Most of the time the knot vector $\mathcal{T}$ will be implicitly defined by the context. In that cases we will write $N_{i,q}(t \mid \mathcal{T})$ just as $N_{i,q}(t)$ for keeping the notation free of unnecessary symbols.
2. For simplicity, we define $N_q(t) := [N_{0,q}(t), \ldots, N_{n-1,q}(t)]^\mathsf{T}$.
3. We define the *collocation matrix* $N_q(\mathbf{T}) := [N_q(t_0), \ldots, N_q(t_{m-1})]^\mathsf{T} \in \mathbb{R}^{n \times m}$ at the sites $\mathbf{T} = [t_0, \ldots, t_{m-1}]^\mathsf{T}$.

By induction one may easily see that B-splines are piecewise polynomials of degree q.

The useful properties of B-spline curves are founded to a great amount on a few essential yet plain observations. Their proof is mostly straightforward and might either be found in [58, pp. 55 – 58] or in [10, pp. 91 – 96].

Theorem 3.5. *(a) Local support: The functions $N_{i,q}(t)$ vanish if $t \notin [\tau_i, \tau_{i+q+1})$.*

(b) For $t \in [\tau_i, \tau_{i+1})$ only the functions $N_{i-q,q}(t), \ldots, N_{i,q}(t)$ may be greater than zero.

(c) Nonnegativity: $N_{i,q}(t) \geq 0$ for all i, q and t.

(d) Partition of one: For $t \in [\tau_i, \tau_{i+1})$ the identity

$$\sum_{j=i-q}^{i} N_{i,q}(t) = 1$$

holds.

As we want to create a class of smooth functions by using linear combinations of B-splines we would expect unique representations for all different possible curves. This is justified by the following, central result:

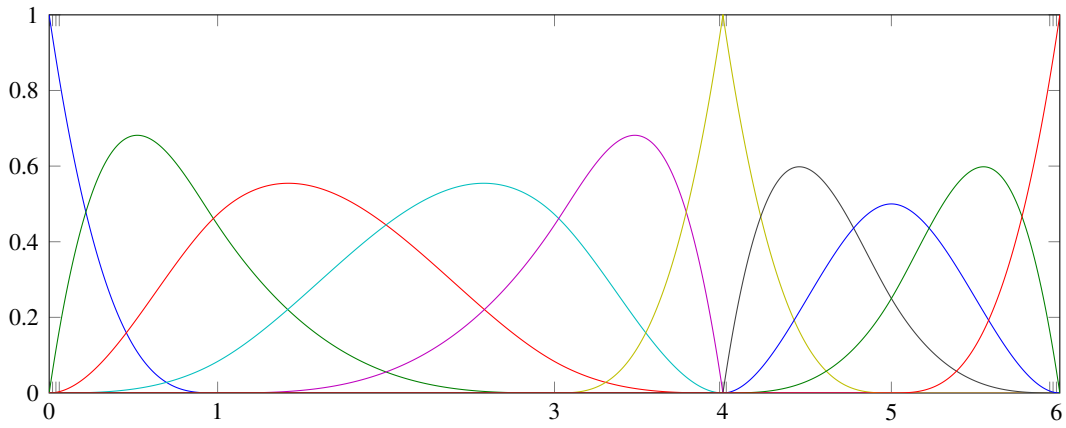


Figure 3.1: The B-splines belonging to the knot vector $\mathcal{T} = [0, 0, 0, 0, 1, 3, 4, 4, 4, 5, 6, 6, 6, 6]$. Note the behaviour at the threefold knot 4.

Theorem 3.6 (Curry & Schoenberg)**.** *For $t \in [\tau_i, \tau_{i+1})$ the functions $N_{i-q,q}, \ldots, N_{i,q}$ are linearly independent.*

A proof may be found in most books on B-splines as well as in numerous books on approximation theory, for example in [21, p. 141].

Finally, we may compute derivatives of the $N_{i,q}$ using the identity

$$N'_{i,q}(t) = \frac{q}{\tau_{i+q} - t_i} N_{i,q-1}(t) - \frac{q}{t_{i+q+1} - t_{i+1}} N_{i+1,q-1}(t). \tag{3.7}$$

Note that one of the divisors again may equal zero. However, as formerly seen with the B-spline recurrence equation, the corresponding B-spline vanishes at the same time [10, p. 117]. Thus, it again is sensible and consistent to define the whole product as zero.

3.2 B-spline curves

Now that B-splines have been introduced, we may finally define spline curves.

Definition 3.8. For a given degree q, a dimension $l \in \mathbb{N}$ and a knot vector $\mathcal{T} = [\tau_0, \ldots, \tau_{n+q}]$ the *spline curve* $f\colon [\tau_0, \tau_{n+q}] \to \mathbb{R}^l$ is defined as

$$f(t) := \sum_{i=0}^{n-1} d_i N_{i,q}(t)$$

with the *control points* $d_i \in \mathbb{R}^l$, $i = 0, \ldots, n-1$.

Convention 3.9. 1. It is desirable to enforce the interpolation of the first and last control point, which means $f(\tau_0) = d_0$ and $f(\tau_{n+q}) = d_{n-1}$. This may be done by ensuring $q+1$-fold multiplicity of the boundary knots [39, p. 106]. In the following we will therefore assume the knot sequence to be of the form

$$\tau_0 = \cdots = \tau_q < \tau_{q+1} \leq \cdots \leq \tau_{n-1} < \tau_n = \cdots = \tau_{n+q}.$$

2. There is no formal requirement on the multiplicity of knots. However, knots with a multiplicity greater than $q + 1$ do not provide further information as this forces B-spline functions to be equal to zero over the whole time interval. Consequently usually one requires the knots to be of maximal multiplicity $q + 1$.

3. The vector space that is spanned by the B-splines of a given knot sequence $\mathcal{T}$ and degree q will be denoted by $\mathbb{S}_q(\mathcal{T})$. According to Theorem 3.6 the functions $N_{0,q}, \dots, N_{n-1,q}$ form a basis of $\mathbb{S}_q(\mathcal{T})$.

Remark 3.10. The representation of spline curves using B-splines is only one of various possible representations, albeit numerically the most stable. We will solely use the form presented and thus not introduce further ones.

We immediately note that most of the properties of B-splines directly carry over to spline curves. For a given point in time t there are at most $q + 1$ of the basis functions greater than zero, thus only $q + 1$ of the control points influence the function value at that specific time. Equally, altering one control point only alters a small part of the whole spline curve.

It is convenient to define $\mathbf{d} = [d_0, \dots, d_{n-1}]^\mathsf{T} \in \mathbb{R}^{n \times l}$ and write the linear combination of B-splines as inner product:

$$\sum_{i=0}^{n-1} d_i N_{i,q}(t) = \mathbf{d}^\mathsf{T} N_q(t). \tag{3.11}$$

Note that the transposition of $\mathbf{d}$ is not necessary if $d_i \in \mathbb{R}$, which is the case most commonly used in this thesis.

One of the most important properties of spline curves is the existence of derivatives. In the open intervals (τ_i, τ_{i+1}) this is guaranteed by the existence of derivatives of the basis functions. However, at the knots a separate result is needed.

Theorem 3.12. *For a spline curve f with knot vector $\mathcal{T}$ the following property holds: If the knot τ_i is of multiplicity k, the function f is $q - k$ times differentiable at τ_i.*

A proof can be found in [40, pp. 55 – 56].

Convention 3.13. In the following we will assume that all inner knots are of multiplicity one. This not only saves us from a bit more complicated discussions when computing derivatives but it also is a sensible choice when modelling the trajectories of vehicles.

Calculating the concrete derivative now may be done by using equation (3.7):

$$\begin{aligned} f'(t) &= \sum_{i=0}^{n-1} d_i N'_{i,q}(t) = \sum_{i=0}^{n-1} d_i \left(\frac{q}{\tau_{i+q} - \tau_i} N_{i,q-1}(t) - \frac{q}{\tau_{i+q+1} - \tau_{i+1}} N_{i+1,q-1}(t) \right) \\ &= q \sum_{i=0}^{n-1} \frac{d_i}{\tau_{i+q} - \tau_i} N_{i,q-1}(t) - q \sum_{i=1}^{n} \frac{d_{i-1}}{\tau_{i+q} - \tau_i} N_{i,q-1}(t) \\ &= q \sum_{i=0}^{n} \frac{d_i - d_{i-1}}{\tau_{i+q} - \tau_i} N_{i,q-1}(t), \end{aligned}$$

where $d_{-1} = d_n := 0$.

While computing the derivatives, we again encounter division by zero for $i = 0$ and $i = n$. However, as before, the corresponding B-splines evaluate to zero as well and hence the corresponding summands can be omitted. Finally, we get

$$f'(t) = q \sum_{i=1}^{n-1} \frac{d_i - d_{i-1}}{\tau_{i+q} - \tau_i} N_{i,q-1}\left(t \mid \mathcal{T}\right) = q \sum_{i=0}^{n-2} \frac{d_{i+1} - d_i}{\tau_{i+q+1} - \tau_{i+1}} N_{i,q-1}\left(t \mid \tilde{\mathcal{T}}\right) \tag{3.14}$$

with $\tilde{\mathcal{T}} = [\tau_1, \ldots, \tau_{n+q-1}]$.

Further derivatives can be computed by repeating that process. It is important to notice that derivatives of spline curves are spline curves as well.

The computation of derivatives may be reduced to matrix multiplication. Let

$$D_n := \begin{bmatrix} -1 & & \\ 1 & \ddots & \\ & \ddots & -1 \\ & & 1 \end{bmatrix} \in \mathbb{R}^{n \times n-1}$$

and

$$\Delta_q \mathcal{T} := \begin{bmatrix} \tau_{q+1} - \tau_1 & & \\ & \ddots & \\ & & \tau_{n+q-1} - \tau_{n-1} \end{bmatrix}^{-1} \in \mathbb{R}^{n-1 \times n-1}.$$

Note that by Convention 3.13 only boundary knots are of multiplicity greater than one. As those boundary knots are left out in the matrix as was done in (3.14), all diagonal elements of $(\Delta_q \mathcal{T})^{-1}$ are greater than zero, hence the matrix is invertible. Then, the derivative may be written as

$$f'(t) = q \mathbf{d}^\mathsf{T} D_n \Delta_q \mathcal{T} N_{q-1}\left(t \mid \tilde{\mathcal{T}}\right).$$

Using the notation presented, derivatives of splines essentially are linear maps of the control points. This will be a valuable tool for spline approximation.

Repeating the process leads to the following, simple observation.

Lemma 3.15. *For a spline curve f of degree q, the knot vector $\mathcal{T} = [\tau_0, \ldots \tau_{n+q}]$, the control point matrix $\mathbf{d}$, the modified knot vectors*

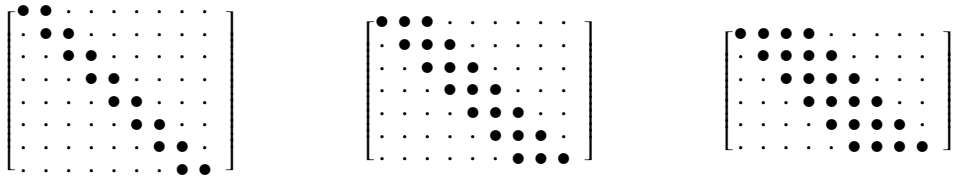
$$\mathcal{T}^{(k)} := [\tau_k, \ldots, \tau_{n+q-k}]$$

and the derivative matrices

$$G_k^\mathsf{T} := \frac{q!}{(q-k)!} \prod_{j=0}^{k-1} D_{n-j} \Delta_{q-j} \mathcal{T}^{(j)},$$

the k-th derivative of f is given by

$$f^{(k)}(t) = \mathbf{d}^\mathsf{T} G_k^\mathsf{T} N_{q-k}\left(t \mid \mathcal{T}^{(k)}\right).$$



(a) G_1 (b) G_2 (c) G_3

Figure 3.2: Sparsity pattern of the first three derivative matrices with $n = 9$.

When dynamics of a function are of interest special attention is paid to the parts where there is no movement, i.e. the function is actually constant. For spline curves constant intervals can be characterized just by looking at the control points. This simple observation will be of key importance in chapter 7.

Theorem 3.16. *A spline curve f of degree q with knot vector $\mathcal{T} = [\tau_i, \dots, \tau_{n+q}]$ and control points $d_0, \dots, d_{n-1}$ is constant over the interval $[\tau_i, \tau_{i+1}]$ if and only if $d_{i-q} = \cdots = d_i$.*

Remark 3.17. Spline curves are piecewise polynomials over knot intervals. Thus, the spline curve is either constant over a whole knot interval or not constant at all. Therefore the restriction on knot intervals in Theorem 3.16 is sufficient.

Proof of Theorem 3.16. Without loss of generality let $t \in (\tau_i, \tau_{i+1})$. For $t = \tau_i$ or $t = \tau_{i+1}$ the result follows by the continuity of spline curves.

If $d_{i-q} = \cdots = d_i$ it follows by the local support and partition of one from Theorem 3.5 that

$$f(t) = \sum_{j=0}^{n-1} d_j N_{j,q}(t) = \sum_{j=i-q}^{i} d_j N_{j,q}(t) = d_i \sum_{j=i-q}^{i} N_{j,q}(t) = d_i.$$

Now we suppose there is another combination of control points $\tilde{d}_0, \dots, \tilde{d}_{n-1}$ such that $f(t) = c$ for all $t \in (\tau_i, \tau_{i+1})$. Then, by using the same properties again,

$$f(t) = \sum_{j=i-q}^{i} c N_{j,q(t)} = \sum_{j=i-q}^{i} \tilde{d}_j N_{j,q}(t)$$

and hence

$$0 = \sum_{j=i-q}^{i} c N_{j,q}(t) - \sum_{j=i-q}^{i} \tilde{d}_j N_{j,q}(t) = \sum_{j=i-q}^{i} (c - \tilde{d}_j) N_{j,q}(t).$$

By the linear independence of the $N_{j,q}$ (Theorem 3.6) it follows that $\tilde{d}_j = c$ for $i = i-q, \dots, i$. □

3.3 Spline evaluation

After defining B-spline curves one major topic is not yet covered: How do we evaluate the values of a B-spline curve in an efficient and numerical stable manner?

The given algorithm manages that in a two step process: First the location of the value to evaluate within the knot sequence is found and then the spline is evaluated using this information.

Searching the correct knot interval can be done by using the well known binary search [17, p. 799].

Algorithm 3.1: Binary search (within a knot sequence)

Input: Knot sequence $\mathcal{T} = [\tau_0, \ldots, \tau_{n+q}]$ for a spline of degree q.
Value $t \in [\tau_q, \tau_n]$ the spline shall be evaluated at.
Result: Index i with $t \in [\tau_i, \tau_{i+1}]$.

```
if t = τ_n then
    return n-1
end if
low ← q
high ← n
mid ← (low + high)/2                          ▷ integer division
while t < τ_mid || t ≥ τ_mid+1 do
    if t < τ_mid then
        high ← mid
    else
        low ←mid
    end if
    mid ← (low + high)/2                      ▷ integer division
end while
return mid
```

As an alternative to binary search we may use plain linear search within the knot interval if some a priori information are available. This situation typically arises when one wants to sample the whole spline function or uses an iterative algorithm that converges to a single value. In both cases the knot interval changes only marginally between different evaluations which suggests using the information available from previous evaluations.

In actual spline implementations it is useful to provide both ways of locating the knot interval and choosing the best suited approach depending on the application.

After the correct interval is found, the algorithm of de Boor may be used to compute the value of the spline and its derivatives. As only convex combinations are used the algorithm offers high numerical stability.

Algorithm 3.2: de Boor [10, pp. 109 – 126]

Input: Spline f with knot sequence $\mathcal{T} = [\tau_0, \ldots, \tau_{n+q}]$ and control points $d_0, \ldots, d_n$.
Value t the spline shall be evaluated at.
Index i such that $t \in [\tau_i, \tau_{i+1})$.

Result: The values $f(t)$, $f'(t)$ and $f''(t)$.

for $i \leftarrow 0$ **to** q **do** ▷ initialization
  $d_m^{(0)} \leftarrow d_{i-m}$
end for

for $j = 1$ **to** q **do**
  for $m = 0$ **to** $q - j$ **do**
    $c \leftarrow \dfrac{t - \tau_{i-m}}{\tau_{i-m+q-j+1} - \tau_{i-m}}$
    $d_m^{(j)} \leftarrow (1 - c) \cdot d_{m+1}^{(j-1)} + c \cdot d_m^{(j-1)}$
  end for
end for

$c_1 \leftarrow \dfrac{q}{\tau_{i+1} - \tau_i}$

$c_2 \leftarrow c_1 \cdot \dfrac{q - 1}{\tau_{i+2} - \tau_i}$

$c_3 \leftarrow c_1 \cdot \dfrac{q - 1}{\tau_{i+1} - \tau_{i-1}}$

$f(t) \leftarrow d_o^{(q)}$

$f'(t) \leftarrow c_1 \cdot \left(d_0^{(q-1)} - d_1^{(q-1)}\right)$

$f''(t) \leftarrow c_2 \cdot d_o^{(q-2)} - (c_3 + c_2) \cdot d_1^{(q-2)} + c_3 \cdot d_2^{(q-2)}$

4 Approximation

In this chapter we will introduce the approximation concepts used throughout this work. While it will come as no surprise that those algorithms will be used in the context of spline approximation all discussion in this chapter will be conducted in a slightly more general form. The actual application to spline approximation problems and all corresponding specializations will be discussed in chapters 6 and 7.

We will denote the m data points to be approximated by $y_0, \dots, y_{m-1} \in \mathbb{R}$ and their corresponding function arguments by $t_0, \dots, t_{m-1}$. In vector notation those will be written as $\mathbf{T} := [t_0, \dots, t_{m-1}]^\mathsf{T}$ and $\mathbf{Y} := [y_0, \dots, y_{m-1}]^\mathsf{T}$.

To approximate a given set of points sensible approaches need to contain a measure ρ of how well the data is approximated. This measure is supposed to assign high values to points that are far away from the approximation and low values to points that are accurately approximated. From a mathematical point of view this requests the function ρ to be *positive and convex* as well as symmetric. A basic approximation problem then may be stated as

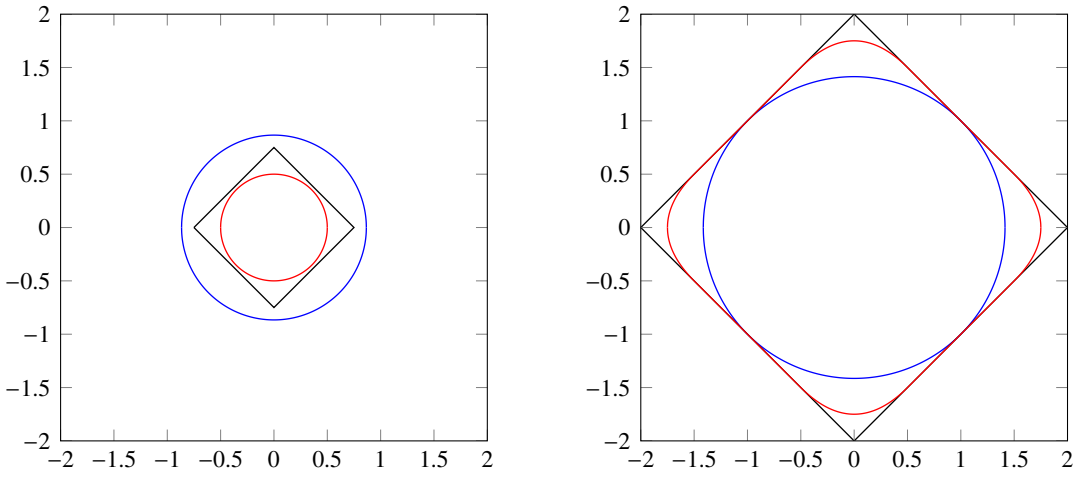
$$\min_f \sum_{i=0}^{m-1} \rho(y_i - f(t_i))$$

where the functions f are chosen from a predefined class of functions. As such arbitrary classes are rather difficult to describe and to deal with mathematically we will only discuss *parametric* approximation. Therefore we make a transition as follows: Instead of our function only being dependent on t_i we introduce the function parameters $x \in \mathbb{R}^n$ and assume that all functions in our defined function class may be written as $f\colon \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$. In the context of spline approximation, i.e. $f \in \mathbb{S}_q(\mathcal{T})$, those functions would be defined as $f(t_i, x) = N_q^\mathsf{T}(t_i)x$ and thus the parameter vector x takes the role of the spline's control points. Using this notation our approximation problem becomes

$$\min_x \sum_{i=0}^{m-1} \rho(y_i - f(t_i, x)). \tag{4.1}$$

In the case of ordinary spline approximation f is linear in the control points x. This, however, will not be enough for all algorithms needed within this thesis. Hence we will also examine cases where the function f is non-linear but still twice continuously differentiable in x.

With $\rho \equiv (\cdot)^2$ this concept of data fitting is well known as *least squares estimation*. The more general form requiring only measurable ρ was proposed as *M-estimator* by Peter Huber in 1967 [43] and since then has been a topic of intensive statistical research. Although we are more interested in the numerical algorithms and properties for solving (4.1), a few distinct choices for ρ stick out from a statistical point of view:



(a) Contours where $\rho(x) + \rho(y) = 0.75$ and $c = 0.5$.

(b) Contours where $\rho(x) + \rho(y) = 2$ and $c = 0.5$.

Figure 4.1: Comparison between the three mentioned choices of ρ. The blue line depicts $\rho \equiv (\cdot)^2$, the black line $\rho \equiv |\cdot|$. The Huber loss function, drawn in red, is a transition between both.

- For $\rho \equiv (\cdot)^2$ the concept is well known and investigated as *least squares approximation*, as has been mentioned above. The estimator (4.1) can be shown to be a maximum likelihood estimator for the function f if the residuals $f(t_i) - y_i$ are normally distributed [55, p. 48][7, p. 4]. If a constant value is fitted (namely $f(t_i, x) \equiv x \in \mathbb{R}$), equation (4.1) is solved by the usual arithmetic mean.
- Setting $\rho \equiv |\cdot|$ yields a so called *least absolute deviations (LAD)* or l_1*-approximation* problem. Then equation (4.1) is a maximum likelihood estimator if the residuals are drawn from a Laplace distribution [22, p. 137]. Fitting a constant value basically is equivalent to finding the median of the data points [8, p. 17].
- Using the Huber loss function

$$\rho(x) = \begin{cases} \frac{x^2}{2c} + \frac{c}{2}, & |x| \leq c, \\ |x|, & |x| > c \end{cases}$$

  a transition between least squares approximation and l_1-approximation is possible. This may be thought as an estimator to the situation in case the errors are assumed to come from a distribution $(1 - \varepsilon)N + \varepsilon E$ where N is normally and E laplacian distributed. There is a direct relationship between the parameters ε and c [26].

A more thorough discussion of the statistical properties of M-estimates and their asymptotics may be found in [44].

Of course other functions ρ have been suggested in literature. It will become clear later on why the given ones are sensible choices for our purpose of data approximation and the available data.

Apart from a clear statistical interpretation those estimators above also offer efficient solution algorithms. In comparison, the least squares solution may be determined significantly faster

than the LAD-solution, with the solution using the Huber loss function ranking somewhere in between. The robustness against outliers on the other hand is greatly increased by the Huber and l_1-solutions. However, depending on the choice of the parameter c there is only little difference between the robustness of those two approaches.

We therefore will concentrate our discussion on the Huber loss function, while we will briefly review the common methods for least squares approximation and will only give a short overview on where to find information how to solve the l_1-problems. The least squares and Huber approach will be utilized in different forms later on.

Up to now we assumed that accuracy is the only aspect that is required for a good approximation of the data. Another important aspect for spline fitting however is that the fitted function should be reasonably smooth. Thus, the addition of a roughness penalty to equation (4.1) may be considered. Without going into to much detail right now, such a roughness penalty may be given as a non-negative function S of the parameter vector x. This rather abstract concept gives us

$$\min_x \sum_{i=0}^{m-1} \rho(y_i - f(t_i, x)) + S(x). \tag{4.2}$$

Finally, a small generalization is possible by adding non-negative weights $w_0, \dots, w_{m-1}$ to the single data points, resulting in the final approximation problem

$$\min_x \sum_{i=0}^{m-1} w_i \rho(y_i - f(t_i, x)) + S(x). \tag{4.3}$$

For simplicity we define $f_i(x) := f(t_i, x)$.

Note that we assumed the function f as well as the values y_i to be one-dimensional. The generalization to use vector valued f can be easily done by defining $\rho^* \colon \mathbb{R}^l \to \mathbb{R}$ as

$$\rho^*(x) := \sum_{i=0}^{l-1} \rho(x_i)$$

with $x = [x_0, \dots, x_{l-1}]^\mathsf{T}$. This interpretation still is consistent with the usual theory, particularly if $\rho \equiv (\cdot)^2$ then $\rho^* \equiv \|\cdot\|_2^2$ and if $\rho \equiv |\cdot|$ then $\rho^* \equiv \|\cdot\|_1$.

4.1 l_2-Approximation

Least squares approximation is a well-known technique used in plenty different contexts and is conceptionally simple as well as thoroughly investigated. Therefore, we will only briefly recall key concepts and reference to standard textbooks for further reading. For example, one may consider [7], [37], [56].

The optimization problem

$$\min_x \sum_{i=0}^{m-1} w_i(y_i - f_i(x))^2 + S(x) \tag{4.4}$$

may be written as

$$\min_x \left\| \sqrt{W}(\mathbf{Y} - f(x)) \right\|_2^2 + S(x) \tag{4.5}$$

with $f(x) = [f_0(x), \ldots, f_{m-1}(x)]$, $W = \operatorname{diag}(w_0, \ldots, w_{m-1})$ and the $\sqrt{\cdot}$ function defined to work component-wise, that is $\sqrt{W} = \operatorname{diag}\left(\sqrt{w_0}, \ldots, \sqrt{w_{m-1}}\right)$.

With $J_f(x)$ being the Jacobian matrix of $f(x)$ it is easy to see that the gradient of the target function is given as

$$2J_f^\mathsf{T}(x)W(\mathbf{Y} - f(x)) + \nabla S(x).$$

It is well known that all minima of a function g will satisfy $\nabla g(x) = 0$ which leads directly to the normal equations of least squares problems with smoothing penalty:

$$J_f^\mathsf{T}(x)Wf(x) + \frac{1}{2}\nabla S(x) = J_f^\mathsf{T}W\mathbf{Y}. \tag{4.6}$$

It is a common situation that the smoothing penalty may be modelled as $S(x) = x^\mathsf{T}Bx$ for some symmetric and positive definite matrix $B \in \mathbb{R}^{n\times n}$. If additionally the f_i are linear, i.e. $f(x) = J_f x$, the normal equations result in a system of linear equations:

$$(J_f^\mathsf{T}WJ_f + B)x = J_f^T W\mathbf{Y}. \tag{4.7}$$

Note that the matrix $J_f^\mathsf{T}WJ_f$ is a Gram matrix and therefore positive semidefinite [38, p. 293]. As $J_f^\mathsf{T}WJ_f + B$ is a sum of positive semidefinite matrices, it is positive semidefinite itself and (4.7) has a solution. This also results in the target function being convex, hence the solution in fact is at the location of the required minimum. If $J_f^\mathsf{T}WJ_f + B$ is positive definite, the solution furthermore is unique.

Solving the problem (4.4) for non-linear f_i gets a bit more interesting. Of course, all general optimization methods are applicable, for example one could solve (4.6) using Newton's method. However, the quadratic structure available might be utilized. A common approach is the Gauß-Newton method that is based on replacing $f_i(x)$ by

$$f_i(x^{(k)} + \delta) \approx f_i(x^{(k)}) + J_f(x^{(k)})\delta.$$

This yields the normal equation

$$J_f^\mathsf{T}(x^{(k)})WJ_f(x^{(k)})\delta + \nabla S(x^{(k)} + \delta) = J_f^\mathsf{T}(x^{(k)})W(\mathbf{Y} - f_i(x^{(k)})) \tag{4.8}$$

that might be used to determine a descent direction. If not available, $\nabla S(x)$ also could be approximated by a Taylor expansion. The value δ that solves equation (4.8) may be used in an additional line search

$$\min_{\alpha\in\mathbb{R}} \left\| \sqrt{W}\big(\mathbf{Y} - f(x^{(k)} + \alpha\delta)\big) \right\|_2^2 + S(x^{(k)} + \alpha\delta)$$

that allows to find a new value $x^{(k+1)} = x^{(k)} + \alpha\delta$.

Note that the Gauss-Newton method is a Quasi-Newton method that uses $J_f^\mathsf{T} J_f$ as an approximation to the Hessian. It can be shown that this method actually converges to a critical point and that the convergence rate is quadratic under reasonable conditions [56, pp. 255 – 257]. Further information might also be found in [7], [37], [63].

If we require even more structure of the problem, further optimizations are possible. We will briefly recall an algorithm that was firstly put to use by Golub and Pereyra [34]. In chapter 7.3 we will use parts of their approach to solve our actual optimization problem efficiently. We will neglect the presence of a smoothing penalty for the discussion at this place. How to include the smoothing penalty as well is discussed in chapter 7.3.

A least squares problem is called *separable* if

$$f\left(\begin{bmatrix} x \\ \xi \end{bmatrix}\right) = J(\xi)x$$

with J being a matrix valued function depending on ξ.

The corresponding separable optimization problem then may be written as

$$\min_{x,\xi} \|\mathbf{Y} - J(\xi)x\|_2^2. \tag{4.9}$$

Now suppose that the optimal value ξ was known as $\hat{\xi}$. In this case the subproblem

$$\min_x \left\|\mathbf{Y} - J(\hat{\xi})x\right\|_2^2$$

is a linear least squares problem and its solution may be found by solving

$$\left(J^T(\hat{\xi})J(\hat{\xi})\right)x = J^\mathsf{T}(\hat{\xi})\mathbf{Y}. \tag{4.10}$$

If $J(\xi)$ has full column rank for all possible values of ξ, inserting the solution in equation (4.9) yields

$$\min_\xi \left\|\left(1 - J(\xi)(J^T(\xi)J(\xi))^{-1}J^\mathsf{T}(\xi)\right)\mathbf{Y}\right\|_2^2. \tag{4.11}$$

In their work Golub and Pereyra use this approach with the Moore-Penrose pseudoinverse instead of (4.10) and thus get rid of the requirement that $J(\xi)$ is of full rank. For us it will be sufficient to stick with $J(\xi)$ of full rank. Recall that ξ is called a *critical point* of f if $\nabla f(\xi) = 0$.

As (4.10) is an orthogonal projection of $\mathbf{Y}$ on the space spanned by $J(\xi)$, Golub and Pereyra named their approach *variable projection algorithm*. They have given an explicit and efficient way to compute the derivative of (4.11) and therefore made it possible to first solve this reduced optimization problem that has been stripped off of all linear variables and just contains the nonlinear part. The full solution then may be found by applying (4.10). Furthermore, they prove the following theorem:

Theorem 4.12 (Golub and Pereyra, 1973 [34])**.** *If $J(\xi)$ has constant rank in an open set Ω around $\hat{\xi}$, the following holds:*

(a) If $\hat{\xi}$ is a critical point of (4.11) *and $\hat{x}$ a solution of equation* (4.10)*, then $[\hat{x}^\mathsf{T}, \hat{\xi}^\mathsf{T}]^\mathsf{T}$ is a critical point of* (4.9)*.*

(b) If $[\hat{x}^\mathsf{T}, \hat{\xi}^\mathsf{T}]^\mathsf{T}$ is a global minimizer of (4.9) *for $\xi \in \Omega$, then $\hat{\xi}$ is a global minimizer of* (4.11) *in Ω.*

In the following let 1_n denote the identity matrix in $\mathbb{R}^n$ where the index n will be omitted if it is clearly defined by the context. We will only introduce the first step of Golub and Pereyra's attack for problem (4.11) under the assumption that $J(\xi)$ has full rank. They compute a QR-decomposition of $J(\xi)$ such that

$$J(\xi) = Q(\xi)R(\xi)P(\xi) = \begin{bmatrix} Q_0(\xi) & Q_1(\xi) \end{bmatrix} \begin{bmatrix} R_0(\xi) \\ 0 \end{bmatrix} P(\xi)$$

with $Q(\xi)$ orthogonal, $R_0(\xi)$ upper triangular and $P(\xi)$ a permutation matrix. Omitting the argument ξ for briefness this yields a rapid simplification of the original problem, as

$$\begin{aligned} J\left(J^\mathsf{T}J\right)^{-1}J^\mathsf{T} &= QRP\left(P^\mathsf{T}R^\mathsf{T}Q^\mathsf{T}QRP\right)^{-1}P^\mathsf{T}R^\mathsf{T}Q^\mathsf{T} \\ &= QR(R^\mathsf{T}R)^{-1}R^\mathsf{T}Q^\mathsf{T} \\ &= Q\begin{bmatrix} R_0 \\ 0 \end{bmatrix}(R_0^\mathsf{T}R_0)^{-1}\begin{bmatrix} R_0^\mathsf{T} & 0 \end{bmatrix}Q^\mathsf{T} \\ &= Q\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} Q_0^\mathsf{T} \\ Q_1^\mathsf{T} \end{bmatrix} = Q\begin{bmatrix} Q_0^\mathsf{T} \\ 0 \end{bmatrix} \end{aligned}$$

and, using $\|Qx\|_2 = \|x\|_2$, it follows that

$$\left\|\left(1 - J(J^\mathsf{T}J)^{-1}J^\mathsf{T}\right)\mathbf{Y}\right\|_2^2 = \left\|Q^\mathsf{T}\left(1 - Q\begin{bmatrix} Q_0^\mathsf{T} \\ 0 \end{bmatrix}\right)\mathbf{Y}\right\|_2^2 = \left\|Q_1^\mathsf{T}\mathbf{Y}\right\|_2^2.$$

This gives another, final reformulation for problem (4.9):

$$\min_\xi \left\|Q_1^\mathsf{T}(\xi)\mathbf{Y}\right\|_2^2. \tag{4.13}$$

Apart from introducing the idea up to this point we will not go too deep into their algorithm. For our purpose it is sufficient to use the theorem above and equation (4.13). We will develop a specialized method depending on this in chapter 7.3. However, for completeness we will give a short overview over the current state of research on variable projection.

Two years after Golub and Pereyra, Linda Kaufman [46] suggested an approximation to the derivative of (4.11) that was easier to compute and still showed similar numerical performance. Later on Ruhe and Wedin [61] showed theoretically that the variable projection approach always converges faster than the same optimization procedure applied to the unreduced problem (4.5). A more modern summary of the algorithm may be found in [7, pp. 351 – 354]. Additionally, Golub and Pereyra summarize the developments of the variable projection method and give an impressive list of applications where the method has been used in [35]. Probably the most recent work has been done by Osborne [57] who explicitly proved convergence rates for the Gauss-Newton algorithm with variable projection.

Remark 4.14. The use of a QR-decomposition to compute the optimal solution of linear least squares problems is numerically more accurate than solving the normal equations (4.7). This is caused by the fact that computing the product $A^\mathsf{T}A$ increases the condition number of A roughly to its square. The QR-decomposition therefore works on a quadratically better conditioned matrix A compared to the normal matrix $A^\mathsf{T}A$.

4.2 Huber-Approximation

In 1964 Peter Huber proposed to use the function

$$\rho_c(x) := \begin{cases} \frac{x^2}{2c} + \frac{c}{2}, & |x| \leq c, \\ |x|, & |x| > c \end{cases} \tag{4.15}$$

as a replacement for the two-norm in data fitting problems [41]. Today this approach is widely accepted as one of the available robust statistical methods. Accordingly, numerous researchers examined different ideas on how to solve those optimization problems efficiently. Usually, one also estimates some scale parameter σ in case it is not known a priorily, resulting in the general optimization problem

$$\min_{x,\sigma} \sum_{i=0}^{m-1} w_i \rho_c \left(\frac{y_i - f_i(x)}{\sigma} \right). \tag{4.16}$$

For linear f_i, Rudolf Dutter provided an in-depth examination of methods to solve similar problems [23]. He proved the convergence of two different methods with different advantages and drawbacks. In the following years it could be shown that both approaches are drawn from the same group of methods which is based on replacing the function ρ_c by a quadratic approximation [16]. Various methods have been proposed that focus on estimating σ alongside x, for example see [71]. Huber's book *Robust Statistics* [44] may be used as a modern reference on this topic.

There have been numerous approaches to find algorithms that are able to solve problem (4.16) for non-linear f_i. Dutter generalized his two algorithms in 1981 [24]. Further work has been done by Dennis [45] as well as Ekblom and Edlund [25], [26].

Obtaining robust smoothing splines by adding a smoothing penalty to (4.16) was suggested by Lenth in 1977 [50] and two years later again by Huber [42]. Utreras [66] suggested an algorithm to compute smoothing splines. Furthermore, there has been done plenty of research on the statistical properties of such smoothing splines, for example see [18], [47], [48]. Yet none of the available publications seems to proof the actual convergence of the straightforward generalization of Huber's and Dutter's algorithms in a general form. We will combine the existing knowledge and augment it such that convergence may be proved under suitable assumptions.

Firstly, we will briefly state some properties of the function ρ_c that help to gain a better understanding. While based on the norm $|\cdot|$, it is immediately clear that ρ_c is no norm anymore, as it is not homogeneous. Some important aspects are preserved though.

Lemma 4.17. *For $x, y \in \mathbb{R}$ and $c > 0$ the function ρ_c*

(a) respects the triangle inequality $\rho_c(x + y) \leq \rho_c(x) + \rho_c(y)$,

(b) fulfils $\rho_c(\alpha x) = |\alpha|\rho_{c|\alpha|}(x)$ *for* $\alpha \in \mathbb{R}$

(c) and is convex.

We observe that (b) is a property similar to homogeneity. The only difference is that the function parameter c also changes under scalar multiplication.

Proof. First note that $|x| \leq \frac{x^2}{2c} + \frac{c}{2}$, as

$$\frac{x^2}{2c} + \frac{c}{2} - |x| = \frac{1}{2c}\left(x^2 - 2c|x| + c^2\right) = \frac{1}{2c}(|x| - c)^2 \geq 0.$$

Hence $|x| \leq \rho(x)$ for $x \in \mathbb{R}$. Now we assume $|x + y| > c$. Then

$$\rho(x + y) = |x + y| \leq |x| + |y| \leq \rho(x) + \rho(y).$$

For $|x + y| \leq c$ the relation $(x + y)^2 = x^2 + 2xy + y^2 \leq c^2$ and therefore $xy \leq \frac{c^2}{2}$ holds. Now for $|x| \leq c$ as well as $|y| \leq c$ we have

$$\rho(x + y) = \frac{(x + y)^2}{2c} + \frac{c}{2} = \frac{x^2}{2c} + \frac{y^2}{2c} + \frac{xy}{c} + \frac{c}{2} \leq \frac{x^2}{2c} + \frac{y^2}{2c} + \frac{c}{2} + \frac{c}{2} = \rho(x) + \rho(y).$$

If either $|x| > c$ or $|y| > c$ then

$$\rho(x + y) \leq \rho(c) = c \leq |x| + |y| \leq \rho(x) + \rho(y),$$

which proves *(a)*.

(b) may be seen by elementary algebraic reformulation and the convexity of ρ immediately follows from the monotonic growth of its derivative

$$\rho_c'(x) = \begin{cases} \frac{x}{c}, & |x| \leq c \\ \operatorname{sgn}(x), & |x| > c. \end{cases} \tag{4.18}$$

□

As the context our problem resides in allows us to provide an estimate of the scale parameter σ, we will neglect the optimization of such a σ and instead hide its value in the parameter c of the function ρ_c, as Lemma 4.17 (b) suggests.

With the addition of the smoothness penalty and by writing

$$r_i(x) := y_i - f_i(x),$$

we have to minimize

$$g(x) := \sum_{i=0}^{m-1} w_i \rho_c(r_i(x)) + S(x). \tag{4.19}$$

The following discussion does not need ρ to be the Huber loss function but works with much weaker assumptions on ρ. Hence we will continue in the general form.

Following Dutter [23] and Byrd [16], the general idea to find a suitable minimum is to replace ρ iteratively by a quadratic approximation q_i at the previous iterations value $\tilde{x}$ such that

$$\rho(r_i(x)) \leq q_i(r_i(x)) \quad \text{for all } x \in \mathbb{R} \tag{4.20}$$

and

$$\rho(r_i(\tilde{x})) = q_i(r_i(\tilde{x})). \tag{4.21}$$

It is obvious that all quadratic functions q_i fulfilling those conditions will be of the form

$$q_i(r_i(x)) = \rho(r_i(\tilde{x})) + \rho'(r_i(\tilde{x}))(r_i(x) - r_i(\tilde{x})) + \frac{1}{2} b_i (r_i(x) - r_i(\tilde{x}))^2. \tag{4.22}$$

The so called *W-algorithm* proposed by Dutter chooses

$$b_i := \begin{cases} \frac{\rho'(r_i(\tilde{x}))}{r_i(\tilde{x})}, & r_i(\tilde{x}) \neq 0, \\ \rho''(0), & r_i(\tilde{x}) = 0. \end{cases}$$

This choice in fact fulfils (4.20) and (4.21) as Dutter has shown:

Lemma 4.23 (Dutter, 1975)**.** *Let ρ be convex and symmetric, $\frac{\rho'(u)}{u}$ bounded and monotonically non-increasing for $u > 0$ and*

$$q(u) := \begin{cases} \rho(\tilde{u}) + \rho'(\tilde{u})(u - \tilde{u}) + \frac{\rho'(\tilde{u})}{2\tilde{u}}(u - \tilde{u})^2, & \tilde{u} \neq 0 \\ \rho(\tilde{u}) + \rho'(\tilde{u})(u - \tilde{u}) + \frac{\rho''(0)}{2}(u - \tilde{u})^2, & \tilde{u} = 0 \end{cases}$$

for some $\tilde{u} \in \mathbb{R}$. Then $\rho(u) \leq q(u)$ and $\rho(\tilde{u}) = q(\tilde{u})$.

As Lemma 4.23 is one of the fundamental pillars for the idea to be presented, we will repeat the proof of Dutter.

Proof. $\rho(\tilde{u}) = q(\tilde{u})$ is trivially true. Similarly for $\tilde{u} = 0$ the result follows immediately by the fact that $\rho(0)$ and $\rho''(0) > 0$. Without loss of generality let $u \neq 0 \neq \tilde{u}$. The difference

$$z(u) := q(u) - \rho(u) = \rho(\tilde{u}) + \rho'(\tilde{u})(u - \tilde{u}) + \frac{\rho'(\tilde{u})}{2\tilde{u}}(u - \tilde{u})^2 - \rho(u)$$

with derivative

$$z'(u) := \rho'(\tilde{u}) + \frac{\rho'(\tilde{u})}{\tilde{u}}(u - \tilde{u}) - \rho'(u) = \frac{\rho'(\tilde{u})}{\tilde{u}} u - \rho'(u)$$

satisfies $z(\tilde{u}) = z(-\tilde{u}) = 0$ as well as $z'(\tilde{u}) = z'(-\tilde{u}) = 0$. Suppose $\tilde{u} > 0$. All that is left to show is $z'(u) \geq 0$ for $u > \tilde{u}$ and $z'(u) \leq 0$ for $0 < u < \tilde{u}$, hence

$$\begin{cases} \frac{\rho'(\tilde{u})}{\tilde{u}} - \frac{\rho'(u)}{u} \geq 0, & u > \tilde{u}, \\ \frac{\rho'(\tilde{u})}{\tilde{u}} - \frac{\rho'(u)}{u} \leq 0, & 0 < u < \tilde{u}, \end{cases}$$

which is true as $\frac{\rho'(u)}{u}$ is monotonically non-increasing. For $\tilde{u} < 0$ the result follows by the symmetry of ρ. □

Byrd proved an even stronger result:

Lemma 4.24 (Byrd, 1979 [16])**.** *Assume ρ satisfies all conditions of Lemma 4.23 and let q be defined as above. Then $q(u) \leq \tilde{q}(u)$ holds for all quadratic functions $\tilde{q}$ with $\rho(u) < \tilde{q}(u)$ and $\rho(\tilde{u}) = \tilde{q}(\tilde{u})$ and arbitrary $u \in \mathbb{R}$.*

Thus, the function class q_i from equation (4.22) with the values b_i as suggested by Dutter is optimal in the sense that there is no other quadratic function as close to ρ that fulfils (4.20) and (4.21). Byrd as well as Wolke and Schwetlick [71] recommended the use of a function that is only locally greater than ρ but offers a better approximation in a neighbourhood of $\tilde{u}$. Though, after numerical testing they come to the conclusion that this is not worth the additional effort in most cases [72]. Similarly, choosing $b_i = \rho''(r_i(\tilde{x}))$ and thus gaining a perfect Taylor approximation also violates the global majoring property (4.20) of q_i that otherwise is present.

As a next step we have to explain how to use those approximations q_i to gain a procedure that actually decreases the value of (4.19). The approach here is nonetheless straightforward and based on the work of Dutter. Let

$$\phi(x) := \sum_{i=0}^{m-1} w_i q_i\big(r_i(x)\big) + S(x).$$

Suppose we can find some $x \in \mathbb{R}$ with

$$\phi(x) < \phi(\tilde{x}).$$

Then according to Lemma 4.23 we have

$$g(x) = \sum_{i=0}^{m-1} w_i \rho(r_i(x)) + S(x) \leq \phi(x) < \phi(\tilde{x}) = g(\tilde{x}). \tag{4.25}$$

Thus, any step decreasing $\phi(\tilde{x})$ also decreases $g(\tilde{x})$.

We have all methods available to optimize $\phi(x)$ as the following lemma shows:

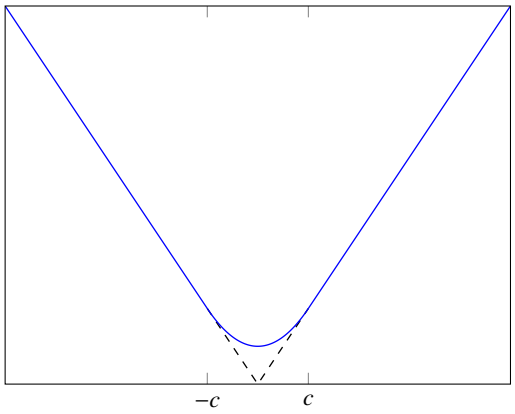
Lemma 4.26. *For symmetric ρ and with*

$$z_i := \begin{cases} \frac{w_i \rho'(r_i(\tilde{x}))}{2 r_i(\tilde{x})}, & r_i(\tilde{x}) \neq 0 \\ \frac{w_i \rho''(0)}{2}, & r_i(\tilde{x}) = 0 \end{cases}$$

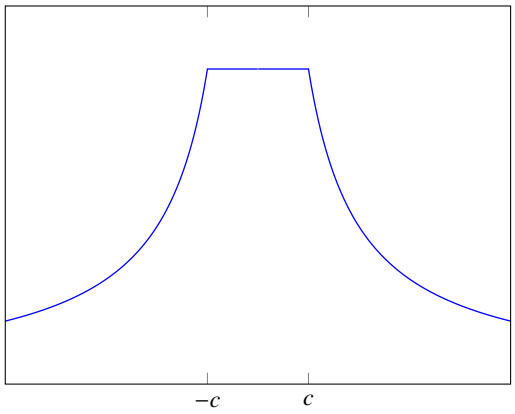
the equation

$$\phi(x) = \sum_{i=0}^{m-1} z_i \big(r_i(x)\big)^2 + S(x) + \kappa$$

with some constant $\kappa \in \mathbb{R}$ holds.



(a) The function ρ_c is just the absolute value function lopped of and completed with a parabola.



(b) The weight z_i plotted as a function of the residual $r_i(x^{(k)})$.

Figure 4.2: The function ρ_c and the corresponding weights z_i.

Proof. The proof follows observations made by Byrd [16]. It is sufficient to examine the single summands. Let $i \in \{0, \dots, m-1\}$. We write $r := r_i(x)$ and $\tilde{r} := r_i(\tilde{x})$. Then for $\tilde{r} \neq 0$

$$\begin{aligned} w_i q_i(r) &= w_i \left(\rho(\tilde{r}) + \rho'(\tilde{r})(r - \tilde{r}) + \frac{\rho'(\tilde{r})}{2\tilde{r}}(r - \tilde{r})^2 \right) \\ &= w_i \left(\rho(\tilde{r}) + \rho'(\tilde{r})(r - \tilde{r}) + \frac{\rho'(\tilde{r})}{2\tilde{r}} r^2 - \rho'(\tilde{r}) r + \frac{1}{2}\rho'(\tilde{r})\tilde{r} \right) \\ &= \frac{w_i \rho'(\tilde{r})}{2\tilde{r}} r^2 + \underbrace{w_i \left(\rho(\tilde{r}) - \frac{1}{2}\rho'(\tilde{r})\tilde{r} \right)}_{=:\kappa_i} = z_i r^2 + \kappa_i \end{aligned}$$

where κ_i does not depend on the variable $r = r_i(x)$. For $\tilde{r} = 0$ we have

$$w_i q_i(r) = w_i \left(\rho(0) + \rho'(0) r + \frac{\rho''(0)}{2} r^2 \right).$$

As ρ is symmetric, $\rho'(0) = 0$ and hence

$$w_i q_i(r) = \frac{w_i \rho''(0)}{2} r^2 + \kappa_i$$

with κ_i constant. □

As the constant κ is not relevant for minimization, $\phi(x)$ is an ordinary least squares problem and therefore may be minimized with the methods discussed in chapter 4.1.

This allows us to formulate a general optimization algorithm for the function g that builds up on any optimization procedure that is able to minimize

$$\sum_{i=0}^{m-1} z_i (r_i(x))^2 + S(x).$$

Algorithm 4.1: Iteratively reweighted least squares

Input: Residual functions $r_i(x)$ and weights w_i for $i = 0, \dots, m-1$.
Roughness penalty $S(x)$.
Initial solution $x^{(0)}$.
Threshold ε.
Algorithm OPTIMSTEP that performs one iteration of an optimization method.

Result: Solution x that minimizes $g(x) := \sum_{i=0}^{m-1} w_i \rho(r_i(x)) + S(x)$.

$k \leftarrow 0$
repeat
  for $i = 0$ **to** $m-1$ **do**
    if $r_i(x^{(k)}) \neq 0$ **then**
      $z_i \leftarrow w_i \frac{\rho'(r_i(x^{(k)}))}{2 r_i(x^{(k)})}$
    else
      $z_i \leftarrow \frac{w_i \rho''(0)}{2}$
    end if
  end for
  $x^{(k+1)} \leftarrow$ OPTIMSTEP($\sum_{i=0}^{m-1} z_i (r_i(x))^2 + S(x)$)
  $k \leftarrow k + 1$
until $g(x^{(k-1)}) - g(x^{(k)}) > \varepsilon$
$x \leftarrow x^{(k)}$

The observation that just the weights of the least squares problem change over the iterations is the reason that this algorithm became known as *Iteratively Reweighted Least Squares* or short *IRLS* in literature.

The following theorem guarantees convergence in a significantly more general form than the previously known results:

Theorem 4.27. *Let ρ be convex and symmetric, $\frac{\rho'(x)}{x}$ bounded and monotonically non-increasing for $x > 0$ and $S(x)$ bounded from below. Suppose that r_i and S are continuously differentiable in a set $\Omega := \{x : g(x) \leq g(x^{(0)})\}$ and the functions ∇r_i^2 as well as ∇S are Lipschitz continuous on Ω. Furthermore denote the function ϕ of the k-th step by ϕ_k. If the optimization procedure used in Algorithm 4.1 is a descent algorithm that sets $x^{(k+1)} = x^{(k)} + \alpha p_k$ for a descent direction p_k with*

$$\frac{-\nabla \phi_k^{\mathsf{T}}(x^{(k)}) p_k}{\|\nabla \phi_k(x^{(k)})\| \, \|p_k\|} \geq \delta > 0 \tag{4.28}$$

for $\|\nabla \phi_k(x^{(k)}\| > 0$ and a step length $\alpha_k > 0$ that satisfies the Wolfe conditions

$$\phi_k(x^{(k+1)}) \leq \phi_k(x^{(k)}) + c_1 \alpha_k \nabla \phi_k(x^{(k)})^{\mathsf{T}} p_k \tag{4.29}$$

$$\nabla \phi_k(x^{(k+1)})^{\mathsf{T}} p_k \geq c_2 \nabla \phi_k(x^{(k)})^{\mathsf{T}} p_k \tag{4.30}$$

for constants $0 < c_1 < c_2 < 1$, then Algorithm 4.1 terminates with

$$\nabla g(x^{(k)}) \to 0.$$

Proof. To prove Theorem 4.27 we will use a common line of argument that for instance also is used in [56, Theorem 3.2]. According to Lemma 4.26

$$\nabla\phi_k(x) = \sum_{i=0}^{m-1} z_i \nabla(r_i(x))^2 + \nabla(S(x)).$$

It follows that $\nabla\phi_k|_\Omega$ is Lipschitz continuous as a linear combination of Lipschitz continuous functions. The weights z_i are bounded above hence there exists a constant L independent of k such that

$$|\nabla\phi_k(x) - \nabla\phi_k(y)| \leq L|x - y|,$$

and, using the Cauchy-Schwarz inequality,

$$\left(\nabla\phi_k(x^{(k+1)}) - \nabla\phi_k(x^{(k)})\right)^\mathsf{T} p_k \leq \alpha_k L \|p_k\|^2.$$

According to equation (4.30) furthermore

$$\left(\nabla\phi_k(x^{(k+1)}) - \nabla\phi_k(x^{(k)})\right)^\mathsf{T} p_k \geq (c_2 - 1)\nabla\phi_k(x^{(k)})^\mathsf{T} p_k.$$

Combining those two equations yields

$$\alpha_k \geq \frac{c_2 - 1}{L} \frac{\nabla\phi_k^\mathsf{T} p_k}{\|p_k\|^2}.$$

Let $\|\nabla\phi_k(x^{(k)}\| > 0$. According to equation (4.29) and Lemma 4.26 each iteration step of the algorithm decreases the value of the current function ϕ_k and according to equation (4.25) also the value of $g(x^{(k)})$. As g is bounded from below $g(x^{(k)}) - g(x^{(k+1)}) \to 0$. Then, again using equation (4.25),

$$0 \leq \phi_k(x^{(k)}) - \phi_k(x^{(k+1)}) \leq g(x^{(k)}) - g(x^{(k+1)}) \to 0.$$

With equation (4.29) and (4.28) it follows that

$$\begin{aligned}
\phi_k(x^{(k)}) - \phi_k(x^{(k+1)}) &\geq -c_1\alpha_k \nabla\phi_k(x^{(k)})^\mathsf{T} p_k \\
&\geq \frac{c_1(c_2 - 1)}{L} \frac{\left(\nabla\phi_k(x^{(k)})^\mathsf{T} p_k\right)^2}{\|p_k\|^2} \\
&\geq \frac{c_1(c_2 - 1)\delta^2}{L} \|\nabla\phi_k(x^{(k)})\| \geq 0.
\end{aligned}$$

and thus $\nabla\phi_k(x^{(k)}) \to 0$. With

$$\begin{aligned}
\frac{\partial}{\partial x_j} g(x^{(k)}) &= \sum_{i=0}^{m-1} \rho'(r_i(x^{(k)})) \frac{\partial}{\partial x_j} r_i(x^{(k)}) + \frac{\partial}{\partial x_j} S(x^{(k)}) \\
&= \sum_{i=0}^{m-1} 2 z_i r_i(x^{(k)}) \frac{\partial}{\partial x_j} r_i(x^{(k)}) + \frac{\partial}{\partial x_j} S(x^{(k)}) = \frac{\partial}{\partial x_j} \phi_k(x^{(k)})
\end{aligned}$$

the result follows. □

Remark 4.31.

1. Theorem 4.27 does not imply convergence of $x^{(k)}$ itself. In fact it is possible that $x^{(k)}$ diverges for certain choices of non-linear r_i. Gaining a procedure that converges globally and unconditionally is not to be expected for arbitrary choices of r_i, though.
2. The properties required by equations (4.28), (4.29) and (4.30) are used solely to ensure proper convergence of the optimization procedure. There exists a multitude of algorithms that guarantee to fulfill those conditions. One may see [56] for further details.
3. The proof of Theorem 4.27 does not rely on (4.28), (4.29) and (4.30) being satisfied in every iteration step. More specific, it is sufficient if there exists K such that the properties are fulfilled for all $k \geq K$.

In chapter 7 we will introduce an algorithm that solves the optimization problem $\min_x \phi_k(x)$ in one step, thus $\min_x \phi_k(x) = \phi_k(x^{(k)})$. This leads us to

Corollary 4.32. *Let ρ be convex and symmetric, $\frac{\rho'(x)}{x}$ bounded and monotonically non-increasing for $x > 0$ and $S(x)$ bounded from below. Suppose that r_i and S continuously differentiable in a set $\Omega := \{x : g(x) \leq g(x^{(0)})\}$ and the functions ∇r_i^2 as well as ∇S are Lipschitz continuous on Ω. Furthermore denote the function ϕ of the k-th step by ϕ_k. If the optimization procedure used in Algorithm 4.1 guarantees $\min \phi_k(x) = \phi_k(x^{(k+1)})$ then Algorithm 4.1 terminates with*

$$\nabla g(x^{(k)}) \to 0.$$

Proof. To use the same line of argument as in the proof of Theorem 4.27 all that we have to show is that for $\phi_k(x^{(k)}) - \phi_k(x^{(k+1)}) \to 0$ it follows that $\nabla \phi_k(x^{(k)}) \to 0$. Suppose $\nabla \phi_k(x^{(k)}) \nrightarrow 0$. Then we could choose an optimization method that guarantees (4.28), (4.29) and (4.30), for example the steepest descent method with an additional line search [56]. This method would result in x' such that $\phi_k(x^{(k)}) \geq \phi_k(x') \geq \phi_k(x^{(k+1)}) = \min_x \phi_k(x)$ and hence $\phi_k(x^{(k)}) - \phi_k(x') \to 0$. The result follows by the same arguments as in the proof of Theorem 4.27. □

In case of the Huber loss function the convergence rate massively depends on the choice of c, as one may easily see by looking at Figure 4.2 (b) on page 37. The bigger c, the less likely it will be that z_i changes for small changes in the residuals. However, if z_i does change only slightly, the approximation of the next step also will be nearly identical. To put it another, rather heuristic way: If the residuals of all but a few data points are smaller than c, the algorithm may solve all of those in one step and needs the iterative procedure just to cope with the outliers.

This chapter will be concluded with a generalization of the convergence result of Dutter that gives an actual lower bound for the decrease of the function g in the most common situation of linear r_i. The presented method closely follows the proof of Dutter [23] and generalizes his version only regarding the presence of a smoothing penalty.

As was mentioned before it is common that $S(x) \equiv x^\mathsf{T} B x$ for symmetric and positive definite B. This results in an optimization problem of the form

$$\min_x \sum_{i=0}^{m-1} \rho(y_i - a_i^\mathsf{T} x) + x^\mathsf{T} B x. \tag{4.33}$$

Writing $\mathbf{Y} := [y_0, \ldots, y_{m-1}]$, $J_f := [a_0, \ldots, a_{m-1}]^\mathsf{T}$ and $Z^{(k)} = \operatorname{diag}(z_0, \ldots, z_{m-1})$, we directly find the solution of

$$\min_x \phi(x) = \sum_{i=0}^{m-1} z_i (y_i - a_i^\mathsf{T} x)^2 + x^\mathsf{T} B x$$

as the vector solving the linear system

$$(J_f^\mathsf{T} Z^{(k)} J_f + B) x^{(k+1)} = J_f^\mathsf{T} Z^{(k)} \mathbf{Y} \tag{4.34}$$

according to chapter 4.1.

Remark 4.35. $J_f^\mathsf{T} Z^{(k)} J_f$ is positive definite as long as J_f is of full rank, since it is a Gram matrix of the matrix

$$\operatorname{diag}\left(\sqrt{(z_0 x^{(k)})}, \ldots, \sqrt{z_{m-1}(x^{(k)})} \right) J_f.$$

However, as $z_i^{(k)}$ may become arbitrary small, it is possible that the linear system is poorly conditioned. For spline approximation, this situation has a clear interpretation: There is a knot interval that only contains outliers but no fairly exact data points. In this case it would be advisable to choose a new knot sequence for the spline approximation containing more data points within that specific interval.

Theorem 4.36. *Given the optimization problem* (4.33)*, a function ρ satisfying the conditions of Theorem 4.27 and the optimization rule* (4.34) *Algorithm 4.1 guarantees*

$$g(x^{(k)}) - g(x^{(k+1)}) \geq (x^{(k+1)} - x^{(k)})^\mathsf{T} (J_f^\mathsf{T} Z^{(k)} J_f + B)(x^{(k+1)} - x^{(k)}) \geq 0$$

in every iteration step.

Proof of Theorem 4.36. Dropping the iteration number (k) from $Z^{(k)}$ and setting $x^{(k)} = \tilde{x}$ and $x^{(k+1)} = x$ for brevity, we may use equation (4.34) and get

$$\begin{aligned}
&g(\tilde{x}) - g(x) \\
&\quad \geq g(\tilde{x}) - \phi(x) \\
&\quad = \sum_{i=0}^{m-1} z_i^{(k)} r_i^2(\tilde{x}) - \sum_{i=0}^{m-1} z_i^{(k)} r_i^2(x) + \tilde{x}^\mathsf{T} B \tilde{x} - x^\mathsf{T} B x \\
&\quad = -2\tilde{x}^T J_f^\mathsf{T} Z \mathbf{Y} + \tilde{x}^\mathsf{T} J_f^\mathsf{T} Z J_f \tilde{x} + 2x^\mathsf{T} J_f^\mathsf{T} Z \mathbf{Y} - x^\mathsf{T} J_f^\mathsf{T} Z J_f x + \tilde{x}^\mathsf{T} B \tilde{x} - x^\mathsf{T} B x \\
&\quad = -2\tilde{x}^\mathsf{T} (J_f^\mathsf{T} Z J_f + B) x + \tilde{x}^\mathsf{T} J_f^\mathsf{T} Z J_f \tilde{x} + 2x^\mathsf{T} (J_f^\mathsf{T} Z J_f + B) x - x^\mathsf{T} J_f^\mathsf{T} Z J_f x + \tilde{x}^\mathsf{T} B \tilde{x} - x^\mathsf{T} B x \\
&\quad = (x - \tilde{x})^\mathsf{T} (J_f^\mathsf{T} Z J_f + B)(x - \tilde{x}).
\end{aligned}$$

□

It may be seen that the presence of a roughness penalty increases the rate of convergence. This side effect is gratefully accepted but was to be expected. Without going into detail, we note that if we decompose $B = C^\mathsf{T} C$, the matrix C takes the role of a *Tikhonov matrix*. As such it is known to decrease the condition number of the original problem [67].

Remark 4.37. In the linear case the optimization problem (4.33) is convex, though not strictly convex. Thus, Algorithm 4.1 not only converges to a critical point but to a global minimum. Since we have no strict convexity, the minimum does not have to be unique.

4.3 A side note to l_1-approximation in the linear case

In the last chapter we introduced Huber's loss function ρ_c and developed an algorithm to solve the corresponding optimization problem. One may note that $\rho_c(\cdot) \xrightarrow{c\to 0} |\cdot|$. So, in a certain sense, solving equation (4.3) with the Huber loss function ρ_c for small c may be seen as an approximation to l_1-approximation. Adcock and Meade [1] explicitly use this algorithm as an attack to classical l_1-approximation problems, probably without being aware of the work of Huber and all publications based on his idea. Another approximation, based on a different weighting scheme, was suggested by Schlossmacher [62].

For linear l_1-problems there furthermore is a vast theory available that does not rely on approximations but is able to solve the problem exactly. We will state the general idea of most of those algorithms just for comparison, but won't dig too deep into the actual methods.

Again suppose that $f_i(x) = a_i^\mathsf{T} x$, $J_f := [a_0, \dots, a_{m-1}]^\mathsf{T}$ and $W := \operatorname{diag}(w_0, \dots, w_{m-1})$. Then the optimization problem may be written as

$$\min_x \sum_{i=0}^{m-1} w_i \left| y_i - a_i^\mathsf{T} x \right| = \min_x \left\| W(\mathbf{Y} - J_f x) \right\|_1.$$

It is well known that with $e = [1, \dots, 1]^\mathsf{T} \in \mathbb{R}^m$ and by introduction of additional variables u, v this problem may be easily written as the linear program

$$\begin{array}{ll} \text{minimize} & e^\mathsf{T} u + e^\mathsf{T} v \\ \text{subject to} & W J_f x + u - v = W\mathbf{Y} \\ & u, v \geq 0. \end{array}$$

For this kind of problem we have some obvious algorithms readily available, the most prominent certainly being the simplex method and interior point algorithms.

The linear l_1-program still provides more structure that can be utilized to design efficient solving algorithms. This has been a topic of intensive research over the past 50 years. A general, slightly outdated overview of theory and algorithms for l_1-approximation is given by Bloomfield and Steiger [8]. To find a suitable minimum, Wagner [68] first suggested transforming those kind of problems to linear programming in 1959, as was done above. Shortly after, Barrodale and Roberts published a modified version of the standard simplex method that efficiently solves the approximation problem [3]–[5]. Various algorithms have been proposed later on, partly based on linear programming, partly leveraging other methods [9], [49]. All those problems, however, have in common that they do not generalize well in cases where a roughness penalty is present or the f_i are not entirely linear.

In terms of performance it is hard to find an objective comparison between the simplex based methods and the approach using the approximation ρ_c. However, Adcock and Meade [1] suggest that the difference is not substantial. While we will not conduct such an examination ourselves, the methods presented above work reasonably fast and compare well to a rough and not fully optimized version of the Barrodale and Roberts algorithm.

The addition of roughness penalties to l_1-problems like we did for the Huber loss problem is not new. Other approaches to attack problems similar to the l_1-problem (4.3) have been suggested,

among others by Bosworth and Lall [13] or Tepper and Sapiro [65]. However, they neither do work well in the given context nor do they generalize nicely to non-linear f_i.

Part III

The approximation of radar traces

5 Identification of standstills

As was discussed previously the identification of standstills is crucial for the whole approximation process. For the human eye it is rather trivial to identify the location of a standstill as we can easily verify by examining Figure 2.2 on page 14. One major rule our mind is using to do this is to judge how far data points are away from each other and to form clusters of points that are close to each other. A well known algorithm that tries to find dense clusters by judging distances between points is called *DBSCAN* (Density based spatial clustering with noise) [28]. The following chapter will introduce that algorithm in a slightly generalized form that fits on the given problem. Afterwards, the concrete application of DBSCAN to the data at hand will be shown and the results will be processed in a way that helps gaining fairly exact standstill intervals.

5.1 Density based spatial clustering with noise

This chapter closely follows the original proposal of DBSCAN by Ester, Kriegel, Sander and Xu [28]. However, they define the algorithm using a given distance metric which actually demands more structure than needed. We will slightly release those requirements and ask only for the properties that are absolutely necessary.

Let $\mathbf{P}$ denote a set of points we want to divide into different clusters. Furthermore we define a predicate $Q\colon \mathbf{P} \times \mathbf{P} \to \{0, 1\}$ describing a concept of closeness, thus $Q(p, q) = 1$ implicates that p and q are close to each other under some notion yet to define. The only requirement of the predicate Q is its symmetry

$$Q(p, q) = Q(q, p) \quad \text{for all } p, q \in \mathbf{P}. \tag{5.1}$$

Intuitively, DBSCAN distinguishes between three different types of points:

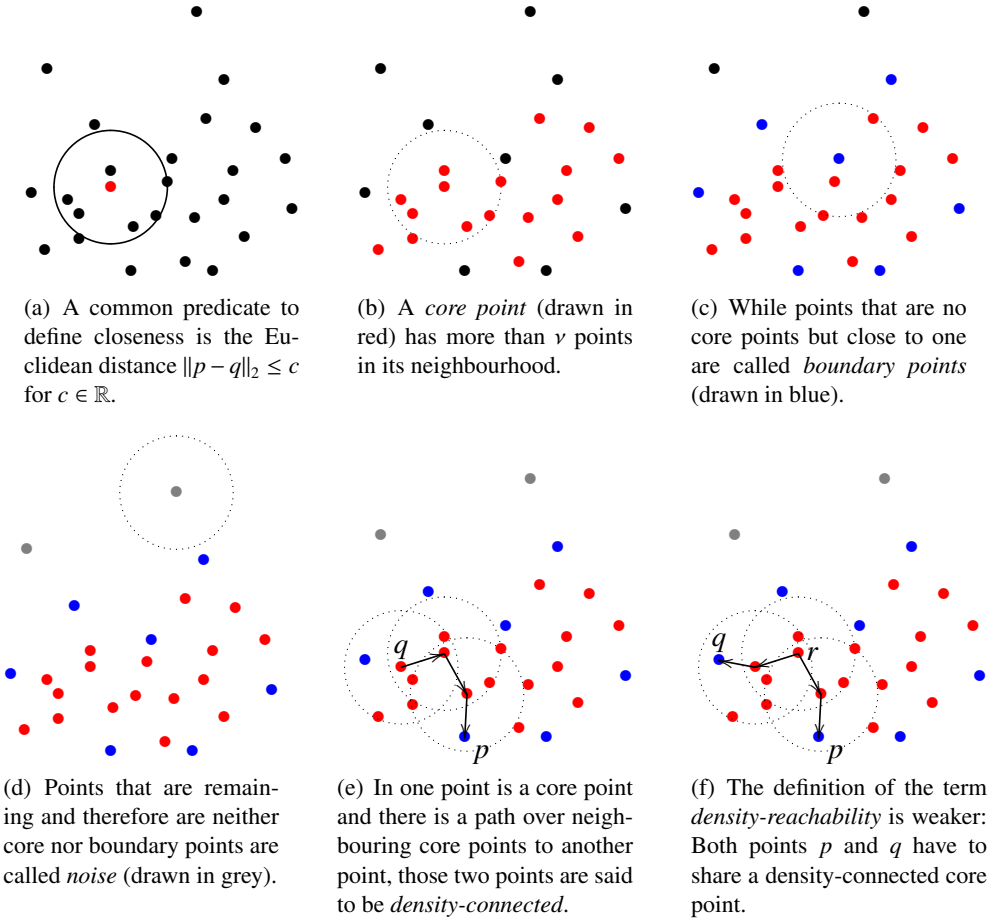
- *Core points* that have a sufficient large amount of points close to them.
- *Boundary points* that are close to a core point but do not have enough neighbours around themselves.
- Points that are neither core nor boundary points are called *noise*.

More formally we require a fair bit of terminology:

Definition 5.2. Let $\mathbf{P}$ be a set of points and $p, q \in \mathbf{P}$.

(a) The *neighbourhood* $\mathcal{N}_Q(p)$ of p with respect to Q is defined as

$$\mathcal{N}_{Q,\mathbf{P}}(p) := \{q \in \mathbf{P} : Q(p, q) = 1\}.$$



(a) A common predicate to define closeness is the Euclidean distance $\|p - q\|_2 \leq c$ for $c \in \mathbb{R}$.

(b) A *core point* (drawn in red) has more than ν points in its neighbourhood.

(c) While points that are no core points but close to one are called *boundary points* (drawn in blue).

(d) Points that are remaining and therefore are neither core nor boundary points are called *noise* (drawn in grey).

(e) In one point is a core point and there is a path over neighbouring core points to another point, those two points are said to be *density-connected*.

(f) The definition of the term *density-reachability* is weaker: Both points p and q have to share a density-connected core point.

Figure 5.1: The concepts behind Definition 5.2 and 5.3 are straightforward to understand visually.

(b) p is called *core point* if $|\mathcal{N}_{Q,\mathbf{P}}| \geq \nu$ for a given value $\nu \in \mathbb{N}$.

(c) p is called *boundary point* if p is not a core point but there is a core point q with $p \in \mathcal{N}_{Q,\mathbf{P}}(q)$.

(d) p is *directly density-reachable* from q if $p \in \mathcal{N}_{Q,\mathbf{P}}(q)$ and q is a core point.

(e) p is *density-reachable* from a core point q if there are core points $p_0, \ldots, p_k$ with $p_0 = q$, $p_i \in \mathcal{N}_{Q,\mathbf{P}}(p_{i-1})$ for all $i = 1, \ldots, k$ and $q \in \mathcal{N}_{Q,\mathbf{P}}(p_k)$.

Density-reachability is transitive. Moreover, using the symmetry of Q, density-reachability is symmetric for core points. As soon as one boundary point is involved, however, the symmetry is lost. Even worse, two boundary points do not have to be density-connected if they share the intuitive property of being in the same neighbourhood. To overcome this limitation we introduce *density-connected* points.

Definition 5.3. Two points p and q are called *density-connected* if there is a core point r such that p as well as q are density-reachable from r.

Density-connectivity in contrast to density-reachability is symmetric even for boundary points. This allows defining the concept of clusters in a mathematical way:

Definition 5.4. Let $\mathbf{P}$ be a set of points. A *cluster* C with respect to the core point size $\nu \in \mathbb{N}$ is a non-empty subset of $\mathbf{P}$ with

(a) If $p \in C$ is a core point and q is density-reachable from p then $q \in C$.

(b) If $p, q \in C$ then p is density-connected to q.

After that terminology-based introduction the definition of the DBSCAN-Algorithm is straightforward:

Algorithm 5.1: DBSCAN [28]

Input: Set $\mathbf{P}$ of points to cluster.
Predicate $Q\colon \mathbf{P} \times \mathbf{P} \to \{0, 1\}$ defining closeness of two points.
Minimum core cluster size ν.
Result: Clusters $C_0, \ldots, C_k$ of $\mathbf{P}$, if those exist.

```
i ← 0                                                    ▷ Cluster index
for all unvisited p ∈ P do
    mark p as visited
    N ← {q ∈ P : Q(p, q) = 1}
    if |N| ≥ ν then                                      ▷ p is a core point
        C_i ← p
        for all p′ ∈ N do
            if p′ is not visited then
                mark p′ as visited
                N′ ← {q ∈ P : Q(p′, q) = 1}
                if |N| ≥ ν then
                    N ← N ∪ N′
                end if
            end if
            if p′ ∉ C_0 ∪ ... ∪ C_{i−1} then
                C_i ← C_i ∪ p′
            end if
        end for
        i ← i + 1
    end if
end for
```

Remark 5.5. It is easy to see that the algorithm in the form stated above has a complexity of $O\left(|\mathbf{P}|^2\right)$. Depending on the nature of Q, it might be possible to define index structures that lower the complexity down to $O(|\mathbf{P}| \log|\mathbf{P}|)$. For example, take the predicate $Q_\delta : \mathbb{R}^2 \times \mathbb{R}^2 \to \{0, 1\}$ with

$$(p, q) \mapsto \begin{cases} 1, & \|p - q\|_2 \leq d, \\ 0, & \text{otherwise.} \end{cases}$$

For this we can sort all points $p \in \mathbf{P}$ with respect to one coordinate direction and determine $p_0, \ldots, p_{m-1}$ such that $p_{x,0} \leq \ldots \leq p_{x,m-1}$. If we now assume a uniform distribution of the values of $p_{x,i}$, the check which points are close to a given point p_i will only require looking

at a neighbourhood $p_{i-j}, \ldots, p_{i+k}$. Assuming that $j + k \ll n$, the dominating task in terms of computational complexity is the sorting algorithm, which could be performed in $O(|\mathbf{P}| \log |\mathbf{P}|)$ steps.

One additional question has to be asked though: Does the order in which the points of $\mathbf{P}$ are visited have an impact on the result? Generally, this is not the case for core points. To proof this, we consider that each point in a cluster C is density-reachable from all core points of the cluster. Therefore, C contains exactly the density-reachable points from an arbitrary core point in C.

In the rare event of two clusters being very close to each other, some boundary points may belong to both. The algorithm as stated would now assign the point to the first cluster identified.

Corollary 5.6 (Correctness of DBSCAN [28])**.** *Core points in the DBSCAN-Algorithm 5.1 are always assigned to the same cluster independently of the processing order.*

5.2 Detection of standstill candidates

The idea behind our approach to identify standstills is simple yet effective: Granted that a target is not moving during a longer period of time, we expect most data points to be within close distance to each other. Finding those clusters of data is exactly what DBSCAN offers to us. All we have to assume is a certain density of data points within a given area.

However, it is immediately clear that slow movements and holding points may be confused by DBSCAN. Thus we will only gain standstill candidates that will have to be refined later on.

There are a few heuristic rules that proved to increase detection accuracy during testing of the methods. While those do not offer deeper mathematical insight, we will outline the entire algorithm for completeness.

Using DBSCAN with the predicate of Remark 5.5 we obtain clusters $C_0, \ldots, C_k$ that form the foundation of our standstill candidates. Coming from those the intervals

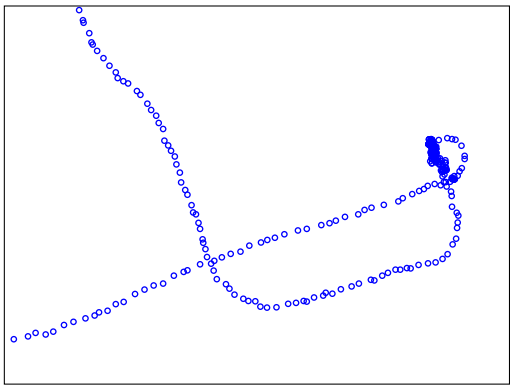
$$S_i := \left[\min\left\{t_j \;\middle|\; p_j \in C_i\right\}, \max\left\{t_j \;\middle|\; p_j \in C_i\right\}\right]$$

are obvious guesses for standstill candidates.

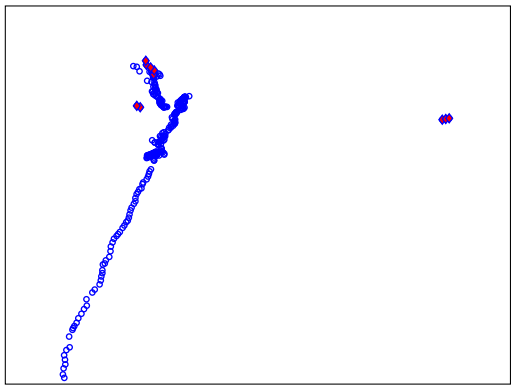
Now it is quite possible that the path of the movement intersects itself. This would result in an area with a high point density. Using the definition of S_i above will yield totally wrong results in this case. The solution is to split up those intervals into multiple ones if a certain amount of consecutive points within the interval does not belong to C_i.

Similarly it is possible that two detected standstill intervals overlap each other. In this case we just might merge both intervals. It has shown to be of advantage to merge standstill candidates if the time between both intervals is shorter than a given value.

Finally, the data quality usually is worst directly at the beginning or end of the recorded data. Because those effects often come along with a standstill, it proved effective to enlarge the standstill to the data boundaries if it is not well separated from those.



(a) Self-intersecting (vehicle) movement. The intersection has a high enough point density to be falsely detected as standstill candidate.



(b) The marked outliers are the first few recorded points of the movement.

Figure 5.2: There are various special cases that might arise after the application of DBSCAN.

We yield the following algorithm:

Algorithm 5.2: Detect standstill candidates

Input: Set of data points $\mathbf{P} \subset \mathbb{R}^2$ and corresponding times $\mathbf{T}$.
Maximum distance between two points $\delta \in \mathbb{R}$ to be close to each other.
Core cluster size $\nu \in \mathbb{N}$.
Maximum gap $\eta \in \mathbb{N}$ within one standstill interval.
Minimum time $\mu \in \mathbb{R}$ between two distinct standstill candidates.
Minimum distance $\sigma \in \mathbb{R}$ from data boundary.

Result: A set $\mathcal{S}$ of standstill candidates $S = [s_0, s_1]$ with $s_0, s_1 \in \mathbf{T}$.

$C_0, \ldots, C_{k-1} \leftarrow \text{DBSCAN}(\mathbf{P}, Q(p,q) := 1_{\|p-q\|_2 \leq \delta}, \nu)$

for $i \leftarrow 0$ **to** $k-1$ **do** ▷ Find standstill candidate intervals.
  $J \leftarrow \{j \mid p_j \in C_i\}$
  $[j_0, \ldots, j_{|C_i|-1}] \leftarrow \text{SORT(J)}$
  $j' \leftarrow j_0$
  for all $i' \in \{i' \mid j_{i'+1} - j_{i'} > \eta\}$ **do**
    $\mathcal{S} \leftarrow \mathcal{S} \cup [t_{j'}, t_{j_{i'}}]$
    $j' \leftarrow j_{i'+1}$
  end for
  $\mathcal{S} \leftarrow \mathcal{S} \cup [t_{j'}, t_{j_{|C|-1}}]$
end for

while there are $S, S' \in \mathcal{S}$ with $\min S' - \max S < \mu$ **do** ▷ Merge overlapping intervals
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S, S'\}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup [\min S, \max S']$
end while

$S \leftarrow \arg\min_{S \in \mathcal{S}} \min S$ ▷ Add boundary points to standstill candidate
if $\min S < \sigma$ **then** $S \leftarrow [t_0, \max S]$

$S \leftarrow \arg\max_{S \in \mathcal{S}} \max S$
if $t_{m-1} - \max S < \sigma$ **then** $S \leftarrow [\min S, t_{m-1}]$

While the algorithm above is heuristically motivated a few simple observations justify its use for standstill detection:

Corollary 5.7. *If $P_i = f(t_i) + \varepsilon_i$ where ε_i are 2-dimensional, independent random vectors and the probability that $\|\varepsilon_i\|_2 \leq \frac{\delta}{2}$ equals p then Algorithm 5.2 finds a standstill period with $k \geq \nu$ recordings with a probability of at least*

$$1 - \sum_{i=0}^{\nu-1} \binom{k}{i} (1-p)^{k-i} p^i.$$

Proof. Take $P_i, P_{i'}$ within the standstill with $\|\varepsilon_i\|_2 \leq \frac{\delta}{2}$ and $\|\varepsilon_{i'}\|_2 \leq \delta$. As $f(t_i) = f(t_{i'})$, it follows that

$$\|P_i - P_{i'}\|_2 = \|\varepsilon_i - \varepsilon_{i'}\|_2 \leq \|\varepsilon_i\|_2 + \|\varepsilon_{i'}\|_2 \leq \delta.$$

Thus P'_i is in the neighbourhood of P_i according to DBSCAN. A cluster and a corresponding standstill candidate hence will be detected if there are ν points P_i with $\|\varepsilon_i\|_2 \leq \frac{\delta}{2}$. On the contrary, it is possible that no standstill is detected if less than ν points are close to $f(t_i)$. The corresponding probability may then be computed by applying the cumulated distribution function of the well known binomial distribution. □

Under perfect conditions we furthermore may show that the standstill candidates overestimate the actual standstill:

Corollary 5.8. *If $t_{i+1} - t_i = 1$ for all $i = 0, \ldots, m-2$ and $P_i = f(t_i)$, $f'(t) = 0$ for $t \in [t_j, t_k]$, $k - j \geq \nu$ and $\|f''(t)\|_2 \leq a$ then the standstill candidate found by Algorithm 5.2 is at least*

$$\sqrt{(i^* - k)^2 + \frac{2\delta}{a}}$$

points too large at each side of the standstill, where

$$i^* := \max_{i>k} \left\{ i : \left\lfloor \sqrt{(i-k)^2 + \frac{2\delta}{a}} \right\rfloor - \left\lceil \sqrt{(i-k)^2 - \frac{2\delta}{a}} \right\rceil \geq \nu - 1 \right\}.$$

Proof. As $k - j \geq \nu$ and $P_j = \ldots = P_k$, the standstill is detected. Let $k \leq i < i'$. Then

$$f(t_{i'}) = f(t_i) + \int_{t_i}^{t_{i'}} f'(t)\,\mathrm{d}t = f(t_i) + \int_{t_i}^{t_{i'}} \left(f'(t_i) + \int_{t_i}^{t} f''(u)\,\mathrm{d}u \right) \mathrm{d}t$$

and therefore

$$\begin{aligned} \|P_i - P_{i'}\|_2 &= \|f(t_i) - f(t_{i'})\|_2 \\ &= \left\| \int_{t_i}^{t_{i'}} \left(f'(t_i) + \int_{t_i}^{t} f''(u)\,\mathrm{d}u \right) \mathrm{d}t \right\|_2 \\ &\leq \int_{t_i}^{t_{i'}} \left(\left\| f'(t_i) \right\|_2 + \left\| \int_{t_i}^{t} f''(u)\,\mathrm{d}u \right\|_2 \right) \mathrm{d}t. \end{aligned}$$

With

$$\left\|f'(t_i)\right\|_2 = \left\|f'(t_k) + \int_{t_k}^{t_i} f''(t)\,\mathrm{d}t\right\|_2 \leq \left\|f'(t_k)\right\|_2 + (t_i - t_k)a = (i-k)a$$

it follows that

$$\begin{aligned}\|P_i - P_{i'}\|_2 &\leq (t_{i'} - t_i)\left\|f'(t_i)\right\|_2 + \int_{t_i}^{t_{i'}} (t - t_i)a\,\mathrm{d}t \\ &\leq (t_{i'} - t_i)(i-k)a + \frac{1}{2}(t_{i'} - t_i)^2 a \\ &= \frac{a}{2}(i' - i)(i + i' - 2k).\end{aligned}$$

Now if

$$\delta \geq \frac{1}{2}(i' - i)(i + i' - 2k)a \geq \|P_i - P_{i'}\|_2, \tag{5.9}$$

then $P_{i'}$ is guaranteed to be in the neighbourhood of P_i. Furthermore

$$\begin{aligned}0 &\geq (i' - i)(i + i' - 2k) - \frac{2\delta}{a} \\ &= i'^2 - 2ki' - i(i - 2k) - \frac{2\delta}{a} \\ &= \left(i' - k + \sqrt{(i-k)^2 + \frac{2\delta}{a}}\right)\left(i' - k - \sqrt{(i-k)^2 + \frac{2\delta}{a}}\right)\end{aligned}$$

which is fulfilled, if

$$|i' - k| \geq \sqrt{(i-k)^2 + \frac{2\delta}{a}}.$$

Assuming $i' > k$, it follows that equation (5.9) is equivalent to

$$i' \leq k + \sqrt{(i-k)^2 + \frac{2\delta}{a}}$$

and, respecting $i' \in \mathbb{N}$

$$i' \leq k + \left\lfloor \sqrt{(i-k)^2 + \frac{2\delta}{a}} \right\rfloor.$$

Similarly, one can show that $P_{i''}$ with $i'' \leq i$ is in the neighbourhood of P_i if

$$i'' \geq k + \left\lceil \sqrt{(i-k)^2 - \frac{2\delta}{a}} \right\rceil.$$

P_i is a core point if $i' - i'' + 1 \geq \nu$, thus the last core point belonging to the cluster of the standstill has the index

$$i^* = \max_{i>k} \left\{ i \;\middle|\; \left\lfloor \sqrt{(i-k)^2 + \frac{2\delta}{a}} \right\rfloor - \left\lceil \sqrt{(i-k)^2 - \frac{2\delta}{a}} \right\rceil \geq \nu - 1 \right\}.$$

The last boundary point of the cluster is the last point in the neighbourhood of this point, which has the index

$$k + \sqrt{(i^* - k)^2 + \frac{2\delta}{a}}.$$

The index of the first point belonging to the cluster before the standstill may be found analogously. □

From a practical point of view Corollary 5.7 and 5.8 offer three important insights:

- If δ is chosen such that $\delta < P_i - f(t_i)$ for most of the recordings, then the probability of detecting all standstills is very high.
- The probability of correct detection increases with the duration of the standstill.
- For reasonably good data the actual standstill is guaranteed to be a subset of the found standstill candidate. Note that we needed strong assumptions to prove Corollary 5.8 mostly for technical reasons. In practice data errors and missing data will change this fact very rarely, if the system is well calibrated. In fact, during testing not one of those situations could be found.

5.3 Isolating standstills

Algorithm 5.2 offers a high probability of standstill detection and is extremely robust against outliers but on the other hand also has a high probability of false-positives as well as a significant overestimation of the standstills duration. It performs especially bad in long periods of slow movements that might be detected as one single standstill period.

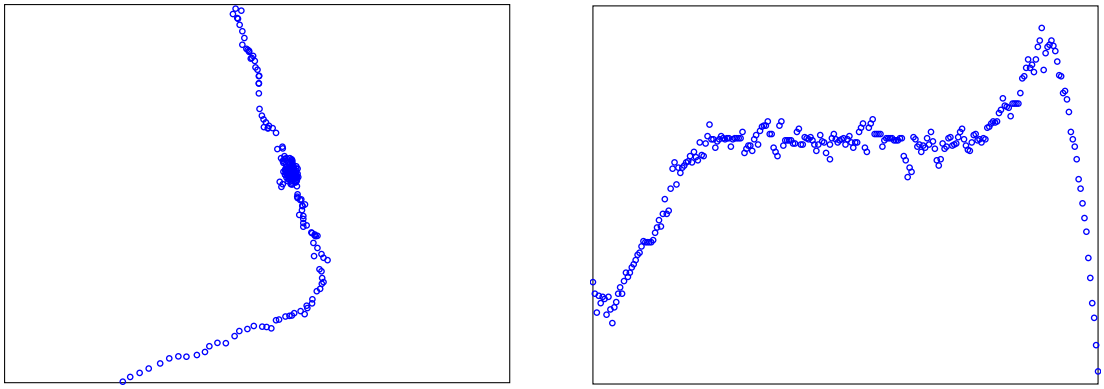
Narrowing down the standstill candidate to fairly accurate standstill positions does require treatment of various special cases. Those are highly situation dependent and do not offer deeper insights. Hence we will restrain ourselves to describing a method that yields reliable results in many cases and will only briefly introduce ideas that might be employed for dealing with the remaining special cases.

Using the concept of core points DBSCAN detects a cluster of points if the point density is high in a variable sized area. For the typical standstill however one has a good estimate on the radius the recorded data points will be distributed around the actual holding position. Thus, with a high probability there is a point p such that $\mathcal{N}_{Q_\delta,\mathbf{P}}(p)$ contains most points belonging to the standstill. We might find such a point by examining

$$p^* := \arg\max_{p \in S_i} |\mathcal{N}_{Q_\delta,S_i}(p)|. \tag{5.10}$$

Firstly, we observe that the proof of Corollary 5.7 is still valid in the current situation. Thus, the detection rate still is high with just random noise. With the parameter d chosen adequately this approach yields a good estimate on the actual standstill location, if it does not exhibit uncommon errors.

However, note that the found point p^* does not have to be unique. We suggest to use the following algorithm that heuristically searches for all plausible standstill locations within the standstill candidate interval S_i.



(a) Projection on the x-y-plane.

(b) The x-coordinate over time.

Figure 5.3: A typical standstill with few outliers and clearly defined standstill location.

Algorithm 5.3: Isolating standstill locations

Input: Points $p_i \in S$ for a standstill candidate S.
Maximum distance $\delta \in \mathbb{R}$ between two points.
Minimum standstill size $\nu \in \mathbb{N}$.

Result: Possible standstill locations $p_0^*, \dots p_k^*$.

$\mathbf{P} \leftarrow S$
$k \leftarrow 0$
while $\max_{p \in P} \left| \mathcal{N}_{Q_\delta, \mathbf{P}}(p) \right| \geq \nu$ **do**
  $p_k^* \leftarrow \arg\max_{p \in P} \left| \mathcal{N}_{Q_\delta, \mathbf{P}}(p) \right|$
  $P \leftarrow P \setminus \mathcal{N}_{Q_\delta, \mathbf{P}}\left(p_k^*\right)$
  $k \leftarrow k + 1$
end while
if $k = 0$ **then** $p_0^* \leftarrow \arg\max_{p \in P} \left| \mathcal{N}_{Q_\delta, \mathbf{P}}(p) \right|$

In the best case the algorithm returns exactly one possible location, p_0^*. Then we may assume that a good guess for our standstill interval is the time spanned by the points within $\mathcal{N}_{Q_\delta, \mathbf{P}}\left(p_k^*\right)$.

Formally we yield

$$\mathbf{T}_k^* := \left\{ t_i \;\middle|\; p_i \in \mathcal{N}_{Q_\delta, \mathbf{P}}\left(p_k^*\right) \right\}.$$

Assuming that $\mathbf{T}_k^*$ might contain the time stamps of two different halts at the same position as well as noise it is possible to use DBSCAN on the set $\mathbf{T}_k^*$. By doing this we can split up different time clusters and remove noise that likely is caused by data outliers that do not belong to the actual standstill.

If $k > 0$ we will have to employ heuristics that allow us to filter out the relevant standstill positions. Without going into to much detail those could include:

- If the time intervals $\mathbf{T}_k^*$, $\mathbf{T}_{k'}^*$ relating to two possible locations are clearly separated we are likely facing two different standstill periods at once. Thus, we might just treat them separately.

- If $\left|\mathbf{T}_k^* \cap \mathbf{T}_{k'}^*\right| \geq j$ for some j both time intervals overlap significantly, we are likely dealing with a standstill that is oscillating between different standstill positions. An example is depicted in Figure 1.2 (f) on page 10. It probably is a good guess to combine both intervals and treat them as one standstill.
- If $\min \mathbf{T}_k^* - \max \mathbf{T}_{k'}^* \leq j$ for some j both time intervals are close to each other. This is the most difficult case as we have to assume that at least one of the time periods is caused by slow movement. If both intervals differ greatly in size, we may just choose the greater one and discard the remaining interval. If, however, both are similar sized, we have to find out if any of them in fact is an actual standstill. One possibility to do this is to fit a straight line through the identified data points and verify whether the slope of this line is close to zero. This would indicate an actual holding position.

If we removed the last line of Algorithm 5.3 we could apply it to raw data points without calling DBSCAN previously. As the requirement to have a lot of points on exactly the same place is a strong one, this does not guarantee that all standstills are found, however. Especially in case of short standstills with a lot of outliers and noise the detection probability of Algorithm 5.2 is noticeably higher.

After the process just described we gain new standstill candidates $S_1, \ldots, S_k$ that should be significantly more precise than the standstill candidates returned by Algorithm 5.2.

To get an even closer estimate on the standstill's location we will make use of a simple observation: After a few seconds of acceleration the typical behaviour around standstills usually can be nicely approximated by a linear function. In particular, for the start time of the standstill S_k we might fit a function

$$t \mapsto \begin{bmatrix} a_x \\ a_y \end{bmatrix} t + \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

such that

$$\sum_{i=i_0}^{i_{\max}} \rho^* \left(\begin{bmatrix} a_x \\ a_y \end{bmatrix} t_{j-i} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} - P_{j-i} \right)$$

is minimal, where j is chosen such that $t_j = \min S_k$. If we compute

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} := \underset{\mathbf{P}_i \in S_k}{\operatorname{median}} \mathbf{P}_i,$$

we gain an estimated standstill start time for each dimension separately by finding t with

$$p_\beta = a_\beta t + b_\beta$$

and adding the previously assumed shift i_0. We yield

$$t_\beta^* := \frac{p_\beta - b_\beta}{a_\beta} + i_0$$

for $\beta \in \{x, y\}$.

As a new beginning of the standstill we choose

$$\min S_k := \max\{t_x^*, t_y^*\}.$$

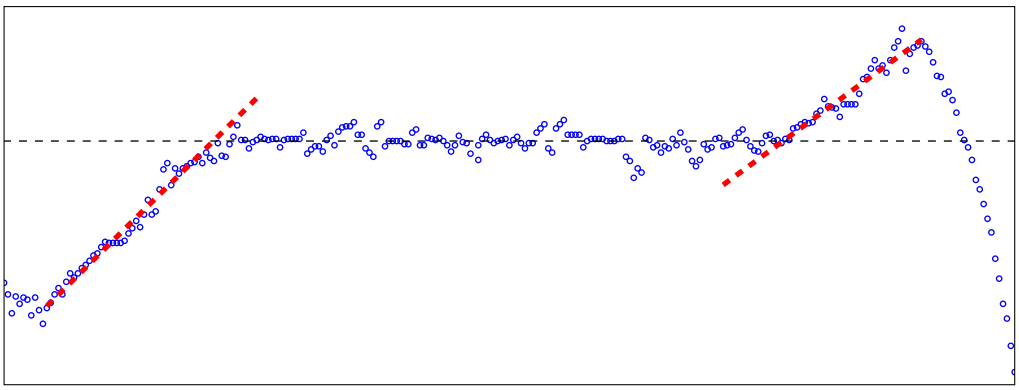


Figure 5.4: The time where the slope before the standstill intersects the standstill's location can be used as an estimated standstill start time. Analogously the standstill end time may be determined.

This procedure can be repeated a few times to gain an optimal standstill location. The end time of the standstill may be treated likewise.

Again, there are various special cases to consider:

- If $|a_\beta|$ is less than a given threshold c for $\beta \in \{x, y\}$, then the values do not change significantly on the corresponding coordinate axis. In this case it is not advisable to use t_β as an estimate for the standstill and we only use the remaining coordinate.
- If $\left\|[a_x, a_y]^\mathsf{T}\right\|_2$ is less than a given threshold c the chosen standstill is too small. We might just enlarge the standstill period and repeat the process.
- If $t_j - t^*_\beta$ is too large, this also indicates a bad fit. One simple solution is to restrict the step in each iteration to only a small change of $\min S_k$.
- It is possible that $\min S_k$ increases over the original ending of the standstill period and $\max S_k$ decreases so that it is smaller than the original beginning. This is a strong sign that the standstill candidate in fact is a false positive detection and we might discard the standstill.

In chapter 2 we mentioned that it is reasonable to enforce the movement directions before and after the standstill to be equal (see Figure 2.3 (e) on page 16). There might be situations where this restriction is not justified by the data. This case can be detected if the angle of the slopes $[a_x, a_y]^\mathsf{T}$ before and after the standstill is significantly greater than zero, thus

$$\frac{a^\mathsf{T} a'}{\|a\|_2 \|a'\|_2} \leq c,$$

if a is the slope before and a' the slope after the standstill.

6 Spline Approximation

Finding a good spline representation for the recorded data points is one of the central problems of this thesis. While we introduced a broad toolkit of approximation techniques in chapter 4, we yet have to transfer the theoretical knowledge to practical spline smoothing problems.

Fitting splines to data usually is a two step process: At first, a knot sequence has to be identified that offers enough flexibility to fit the required curve but at the same time minimizes storage requirements. Afterwards, the concrete values of the control points have to be determined. If necessary, those steps may be iterated to incorporate some kind of adaptive algorithm that modifies the knot sequence depending on the accuracy and curvature of a previous fit.

For the moment we will neglect the problem of finding good knot sequences and assume $\mathcal{T}$ to resemble an a priori given knot sequence whenever needed. We will discuss how to choose a good knot sequence in chapter 8.2.

Spline approximation in theory is possible for data points of arbitrary dimension - we will see that problems in multiple dimensions might just be reduced to multiple one-dimensional problems.

As it was done in chapter 3.2, we will denote the $n \times l$-matrix containing the control points of a spline curve by $\mathbf{d}$. If $l \in \{2, 3\}$, we furthermore use $\mathbf{d}_x$, $\mathbf{d}_y$ and, if necessary, $\mathbf{d}_z$ for the single columns of $\mathbf{d}$.

Up to now we considered the addition of a roughness penalty in a very abstract form (see eq. (4.2)). With the concrete concept of smooth functions for approximation in mind, it is a reasonable approach to interpret smoothness as change in one of the derivatives, preferably the second. This brings us to

$$\min_{f \in \mathbb{S}_q(\mathcal{T})} \sum_{i=0}^{m-1} w_i \rho^*(P_i - f(t_i)) + \lambda \int_{\mathbb{R}} \left\| f''(x) \right\|_2^2 \mathrm{d}x \tag{6.1}$$

with $\lambda > 0$. The *smoothing parameter* λ may be used to influence the balance between approximation error and smoothness. With $\rho^* \equiv \|\cdot\|_2^2$ this concept is commonly called *smoothing spline*. Using the Huber loss function for ρ^* we gain a *robust smoothing spline*.

It is important to notice that – in contrast to equation (4.1) – the vector within the norm might be multidimensional. However, as we defined ρ^* to be additive in its components we just may solve each dimension separately and solve the approximation problems

$$\min_{f_\beta \in \mathbb{S}_q(\mathcal{T})} \sum_{i=0}^{m-1} w_i \rho\left(P_{\beta,i} - f_\beta(t_i)\right) + \lambda \int_{\mathbb{R}} \left(f_\beta''(x)\right)^2 \mathrm{d}x, \qquad \beta \in \{x, y, z\}.$$

Allowing the smoothing parameter λ to vary over time is a small generalization that permits much more flexible modelling, because it enables varying accuracy in the data to be respected. We yield the approximation problems

$$\min_{f_\beta \in \mathbb{S}_q(\mathcal{T})} \sum_{i=0}^{m-1} w_i \rho\left(P_{\beta,i} - f_\beta(t_i)\right) + \int_{\mathbb{R}} \lambda(x) \left(f_\beta''(x)\right)^2 \,\mathrm{d}x, \qquad \beta \in \{x, y, z\}. \tag{6.2}$$

From chapter 4.2 we know that this problem may be attacked using the IRLS Algorithm 4.1 that requires solving multiple least squares problems

$$\min_{f_\beta \in \mathbb{S}_q(\mathcal{T})} \sum_{i=0}^{m-1} w_i \left(P_{\beta,i} - f_\beta(t_i)\right)^2 + \int_{\mathbb{R}} \lambda(x) \left(f_\beta''(x)\right)^2 \,\mathrm{d}x, \qquad \beta \in \{x, y, z\}. \tag{6.3}$$

Writing $f_\beta(t_i) = N_q^\mathsf{T}(t_i)\mathbf{d}_\beta$ as in (3.11) and

$$f_\beta''(x) = N_{q-2}^\mathsf{T}\left(x \mid \mathcal{T}^{(2)}\right) G_2 \mathbf{d}_\beta$$

using the derivative matrices of Lemma 3.15, problem (6.3) may be transformed to

$$\min_{\mathbf{d}_\beta \in \mathbb{R}^n} \sum_{i=0}^{m-1} w_i \left(P_{\beta,i} - N_q^\mathsf{T}(t_i)\mathbf{d}_\beta\right)^2 + \int_{\mathbb{R}} \lambda(x) \left(N_{q-2}^\mathsf{T}\left(x \mid \mathcal{T}^{(2)}\right) G_2 \mathbf{d}_\beta\right)^2 \,\mathrm{d}x, \qquad \beta \in \{x, y, z\} \tag{6.4}$$

for given weights w_i and $\lambda\colon [\tau_0, \tau_{n+q}] \to \mathbb{R}$.

Apart from the roughness penalty the optimization problem clearly is quadratic. We furthermore have

$$\begin{aligned}
&\int_{\mathbb{R}} \lambda(x) \left(N_{q-2}^\mathsf{T}(x \mid \mathcal{T}^{(2)}) G_2 \mathbf{d}_\beta\right)^2 \,\mathrm{d}x \\
&= \int_{\mathbb{R}} \lambda(x) \mathbf{d}_\beta^\mathsf{T} G_2^\mathsf{T} N_{q-2}(x \mid \mathcal{T}^{(2)}) N_{q-2}^\mathsf{T}(x \mid \mathcal{T}^{(2)}) G_2 \mathbf{d}_\beta \,\mathrm{d}x \\
&= \mathbf{d}_\beta^\mathsf{T} G_2^\mathsf{T} \left(\int_{\mathbb{R}} N_{q-2}(x \mid \mathcal{T}^{(2)}) \lambda(x) N_{q-2}^\mathsf{T}(x \mid \mathcal{T}^{(2)}) \,\mathrm{d}x\right) G_2 \mathbf{d}_\beta.
\end{aligned}$$

This yields a *Gram matrix*

$$B_q(\mathcal{T}, \lambda) := \int_{\mathbb{R}} N_{q-2}(x \mid \mathcal{T}^{(2)}) \lambda(x) N_{q-2}^\mathsf{T}(x \mid \mathcal{T}^{(2)}) \,\mathrm{d}x \tag{6.5}$$

which is positive semidefinite as long as $\lambda(x) \geq 0$ for all $x \in \mathcal{T}$.

The computation of the matrices $B_q(\mathcal{T}, \lambda)$ is completely possible for arbitrary λ. The computational effort, however, is much smaller if we restrict ourselves to the case when λ is piecewise constant over the knot sequence $\mathcal{T}$. Then

$$\lambda(t) = \sum_{i=q}^{n-1} \lambda_i 1_{[\tau_i, \tau_{i+1})}(t)$$

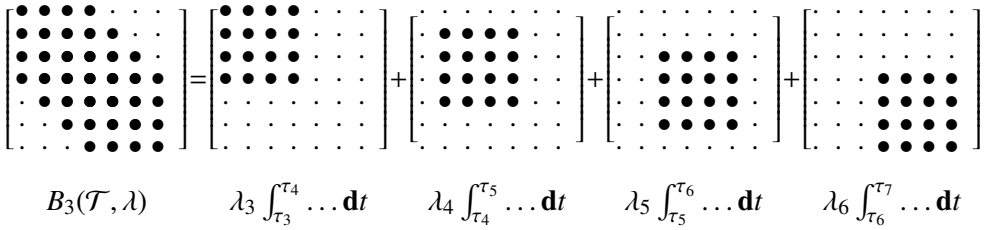


Figure 6.1: Sparsity pattern of the matrix $B_3(\mathcal{T}, \lambda)$ and its summands with $n = 7$.

for some $\lambda_i \in \mathbb{R}$. The smoothing parameter λ is not a direct part of the function to approximate, but rather a parameterization parameter that can be used for generating the best fitting curve. Hence this simplification does not reduce the freedom for modelling the curve in a significant way. Nonetheless, it reduces the computational overhead induced by varying λ over time to matrix addition and scalar multiplication.

Now we note that using the locality of splines (3.5 (a)) the Gram matrix can be written as a sum of integrals.

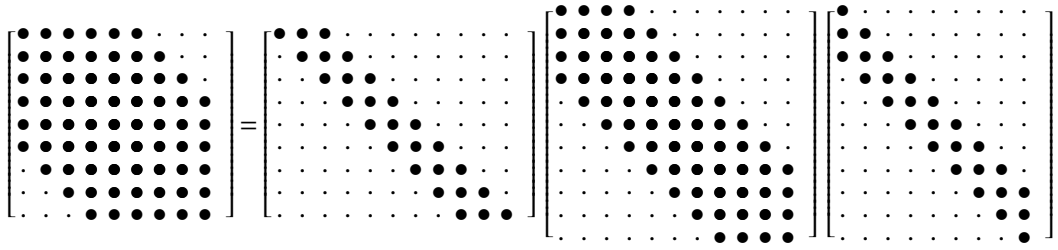
$$\begin{aligned} B_q(\mathcal{T}, \lambda) &= \int_{\mathbb{R}} N_{q-2}(x \mid \mathcal{T}^{(2)}) \lambda(x) N_{q-2}^{\mathsf{T}}(x \mid \mathcal{T}^{(2)}) \, \mathrm{d}x \\ &= \sum_{i=q}^{n-1} \lambda_i \int_{\tau_i}^{\tau_{i+1}} N_{q-2}(x \mid \mathcal{T}^{(2)}) N_{q-2}^{\mathsf{T}}(x \mid \mathcal{T}^{(2)}) \, \mathrm{d}x. \end{aligned}$$

Using Gaussian quadrature of degree $q-1$ (see Appendix A.1), we now can compute the integrals exact up to rounding errors. This is possible because the integrand is just a matrix of polynomials of degree $2(q-2)$.

Remark 6.6. When actually computing the matrix $B_q(\mathcal{T}, \lambda)$, a few simplifications can be made:

1. It is not necessary to deal with the modified knot sequences $\mathcal{T}^{(2)}$ since shifting indices yields $N_{q-2}(x \mid \mathcal{T}^{(2)}) = [N_{2,q-2}(x \mid \mathcal{T}), \dots, N_{n+2-1,q-2}(x \mid \mathcal{T})]^{\mathsf{T}}$.
2. As within a given knot interval only a few splines are greater than zero (Theorem 3.5 (b)), the matrix to be integrated is nonzero only at the rows and columns with index $i-q, \dots, i$. Furthermore, that matrix as well as $B_q(\mathcal{T}, \lambda)$ is symmetric.
3. If some iterative algorithm is used to find good values for the λ_i, it is possible to save only the nonzero submatrices that are summed up weighted by lambda. That way, recomputing $B_q(\mathcal{T}, \lambda)$ boils down to matrix addition and scalar multiplication.

Remark 6.7. Not only the summands, but also the matrix $B_q(\mathcal{T}, \lambda)$ is sparse. Only the diagonal and q sub-and superdiagonals are filled. Due to the sparsity of the matrix G_k the product $G_k^{\mathsf{T}} B_q(\mathcal{T}, \lambda) G_k$ also is banded. Only $q + k$ sub- and superdiagonals are filled, as one can see in Figures 6.1 and 6.2.



$G_2^\mathsf{T} B_3(\mathcal{T}, \lambda) G^2$ G_2^T $B_3(\mathcal{T}, \lambda)$ G_2

Figure 6.2: Sparsity pattern of the matrix $G_2^\mathsf{T} B_3(\mathcal{T}, \lambda) G_2$ with $n = 11$. Only the diagonal and $q+k$ sub- and superdiagonals are filled.

In chapter 7.3 it will be useful to decompose $B_q(\mathcal{T}, \lambda) = C^\mathsf{T} C$. Using any exact quadrature rule for integration, we may find weights ω_k and points $S = [s_k]$ such that

$$\begin{aligned} B_q(\mathcal{T}, \lambda) &= \left[\int_{\mathbb{R}} N_{i,q-2}(x \mid \mathcal{T}^{(2)}) \lambda(x) N_{j,q-2}(x \mid \mathcal{T}^{(2)}) \, \mathrm{d}x \right]_{i,j=0,\dots,n} \\ &= \left[\sum_k \omega_k \lambda(s_k) N_{i,q-2}(s_k \mid \mathcal{T}^{(2)}) N_{j,q-2}(s_k \mid \mathcal{T}^{(2)}) \right]_{i,j=0,\dots,n} \\ &= \left[N_{i,q-2}^\mathsf{T}(S \mid \mathcal{T}^{(2)}) \Omega N_{j,q-2}(S \mid \mathcal{T}^{(2)}) \right]_{i,j=0,\dots,n} \\ &= N_{q-2}(S \mid \mathcal{T}^{(2)}) \Omega N_{q-2}^\mathsf{T}(S \mid \mathcal{T}^{(2)}), \end{aligned}$$

with

$$\Omega = \mathrm{diag}([\omega_k \lambda(s_k)]).$$

Altogether this gives us

$$\int_{\mathbb{R}} \lambda(x) \left(N_{q-2}^\mathsf{T}\left(x \mid \mathcal{T}^{(2)}\right) G_2 \mathbf{d}_\beta \right)^2 \, \mathrm{d}x = \mathbf{d}_\beta^\mathsf{T} G_2^\mathsf{T} N_{q-2}(S \mid \mathcal{T}^{(2)}) \Omega N_{q-2}^\mathsf{T}(S \mid \mathcal{T}^{(2)}) G_2 \mathbf{d}_\beta \tag{6.8}$$

for $\beta \in \{x, y, z\}$.

Now, the solution to the least squares variant of the smoothing spline problem (6.4) is a direct application of the theory from chapter 4.1:

$$\left(N_q^\mathsf{T}(\mathbf{T}) W N_q(\mathbf{T}) + G_2^\mathsf{T} B_q(\mathcal{T}, \lambda) G_2 \right) \mathbf{d}_\beta = N_q^\mathsf{T}(\mathbf{T}) W \mathbf{P}_\beta, \qquad \beta \in \{x, y, z\}. \tag{6.9}$$

Note that if we solve the problem for multiple β we may reuse the matrix decomposition computed for solving (6.9) and get the second and third column of $\mathbf{d}$ cheaply.

The solution using the Huber loss function may be determined by using Algorithm 4.1 with the optimization step:

$$\left(N_q^\mathsf{T}(\mathbf{T}) Z_\beta^{(k)} N_q(\mathbf{T}) + G_2^\mathsf{T} B_q(\mathcal{T}, \lambda) G_2 \right) \mathbf{d}_\beta = N_q^\mathsf{T}(\mathbf{T}) Z^{(k)} \mathbf{P}_\beta, \qquad \beta \in \{x, y, z\}. \tag{6.10}$$

Because of the weight matrix $Z_\beta^{(k)}$ depending on β, it will not be possible to reuse the matrix decomposition in this case though.

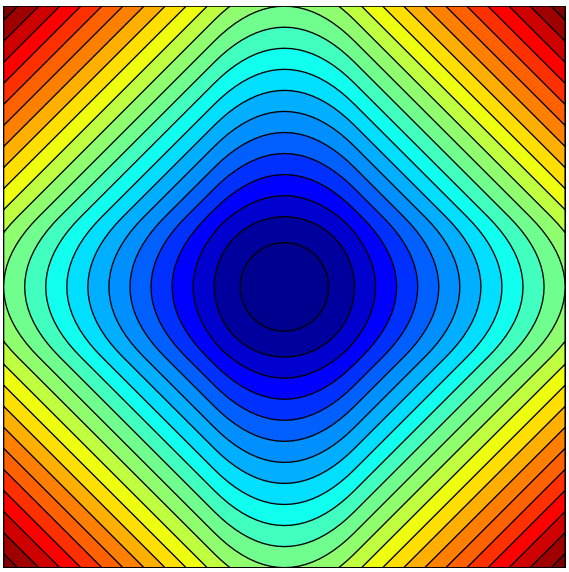


Figure 6.3: Looking at the contours of ρ_c^* with $l = 2$ one may easily see that it is invariant under rotations only for values with small euclidean norm.

Convention 6.11. As long as $\mathbf{T}, \mathcal{T}$ and λ are clearly defined by the context we will write N instead of $N_q(\mathbf{T})$ and just B instead of $G_2^\mathsf{T} B_q(\mathcal{T}, \lambda) G_2$ in the following.

Remark 6.12. There is one important difference between approximation with the Huber loss function ρ_c and ordinary least squares approximation that only stands out when looking at the multidimensional case. Given an arbitrary rotation matrix R the relation

$$\sum_{i=0}^{m-1} w_i \rho^* \left(P_i - d^\mathsf{T} N_q(t_i) \right) = \sum_{i=0}^{m-1} w_i \rho^* \left(RP_i - Rd^\mathsf{T} N_q(t_i) \right)$$

will in general hold only if $\rho^*(Rx) = \rho^*(x)$ for all $x \in \mathbb{R}^l$. This is true for $\|\cdot\|_2^2$, however one may easily see that generally $\|Rx\|_1 \neq \|x\|_1$ as well as $\overline{\rho}(Rx) \neq \overline{\rho}(x)$ with ρ being the Huber loss function. Therefore, our robust approximation techniques are not invariant under rotations. This is a small drawback. However, especially when approximating spatial data, the advantages of robust approximation outweighs the disadvantage for the problem at hand.

As the Huber loss function uses the two norm for small residuals, we yield an effect that might be called *partially invariant under rotations*. If all errors are small compared to the parameter c of the loss function, the result will be invariant under rotations while the influence of outliers changes depending on the chosen coordinate axes. This effect is also clearly visible in Figure 6.3.

7 Approximation of standstill periods

After identifying standstill candidates and discussing how general spline approximation works, we have to take care of adding the boundary conditions that we identified as useful in chapter 2 to the approximation algorithm.

Therefore, we will temporarily assume that the vehicle whose movement is to be approximated is known to have been standing still in the time interval $I \subset \mathcal{T}$. Furthermore, as standstills should not happen during flight of aircrafts, we will also ignore the height coordinate and assume $l = 2$ for the whole chapter.

To gain a correct approximation, this fact has to be enforced by our approximation procedure. The first, intuitive condition is that the approximation actually is constant over these time intervals, thus

$$f'(t) = 0 \qquad \text{for all } t \in I. \tag{7.1}$$

We furthermore recognized it to be desirable that the direction in which the holding position is reached matches the direction it is left. This might be modelled with the condition

$$f'(\min I - \varepsilon) \text{ and } f'(\max I + \varepsilon) \text{ are linearly dependent for small } \varepsilon. \tag{7.2}$$

If the standstill is known to be in a parking area we might have additional information available that tells us the angle in which the vehicle or aircraft is supposed to enter or leave its parking position. Hence, if an intended movement direction $\delta = [\delta_x, \delta_y]^\mathsf{T} \in \mathbb{R}^2$, $\|\delta\|_2 = 1$ is available, we yield a replacement for (7.2):

$$\begin{aligned} f'(\min I - \varepsilon) &= \mu\delta \qquad \text{for some } \mu \in \mathbb{R}, \\ f'(\max I + \varepsilon) &= \lambda\delta \qquad \text{for some } \lambda \in \mathbb{R}. \end{aligned} \tag{7.3}$$

Clearly, (7.1) will always have to be fulfilled while (7.2) and (7.3) depend on the actual situation and information available. In the following we will develop methods to accommodate those conditions into our approximation concept. We will start with the most simple condition (7.1) and successively add further conditions.

For our purpose it is sufficient to assume that there is only one standstill period in the time interval to be approximated.

7.1 Enforcing $f'(t) = 0$

We will start the approximation process by deciding on a knot sequence that is suitable for the standstill. By Remark 3.17 we know that the spline curve has to be constant over a whole knot interval if it is constant at all. Thus, we choose our first two knots τ_k, τ_{k+1} such that $I = [\tau_k, \tau_{k+1}]$. How we build the knot sequence around those knots is not too important for the further algorithm and rather a topic of parameterization. If the data is equally distributed we could for example choose

$$\mathcal{T} = [\ldots, \tau_k - 3c, \tau_k - 2c, \tau_k - c, \tau_k, \tau_{k+1}, \tau_{k+1} + c, \tau_{k+1} + 2c, \tau_{k+1} + 3c, \ldots,]$$

for some constant c. In this case, c should be chosen in a way that each knot interval contains at least one, better multiple data points. The bigger c, the less flexible the spline curve will be. If the standstill touches the boundary of the recorded data, we might just expand the knot sequence to one direction and insert the other delimiter of the standstill period as a q-fold boundary knot.

According to Theorem 3.16 condition (7.1) may be enforced by requiring

$$d_{k-q} = \cdots = d_k. \tag{7.4}$$

In matrix notation this yields the linear boundary conditions

$$Ed_\beta = 0, \qquad \beta \in \{x, y\}$$

with

$$E := [0_{q,k-q-1}, \overline{E}, 0_{q,n-k}] \in \mathbb{R}^{q \times n},$$

and

$$\overline{E} := \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{q,q+1}.$$

This could be incorporated into the optimization problem as linear boundary conditions, for example using Lagrange multipliers. However, this approach, as it is described by Nocedal and Wright, has the disadvantage that the resulting matrix for the system of linear equations can be shown to be indefinite [56, p. 454]. Therefore it may not be solved as fast as it would be possible using positive definite matrices.

A simple and totally different attack is to substitute $\mathbf{d}_\beta = Kx_\beta$ with K chosen such that $EK = 0$. If K is a basis of the kernel of E, obviously those and only those $\mathbf{d}_\beta$ that can be represented by Kx_β fulfil the boundary constraints.

As the structure of E is relatively simple, a basis of its kernel may be explicitly given as

$$K := \begin{bmatrix} 1_{k-q-1} & & \\ & \overline{K} & \\ & & 1_{n-k} \end{bmatrix} \in \mathbb{R}^{n \times n-q},$$

and

$$\overline{K} := \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{q+1}.$$

This way, the approximation problem (6.4) becomes

$$\min_{x_\beta \in \mathbb{R}^{n-q}} \sum_{i=0}^{m-1} w_i \left(P_{\beta,i} - N_q^\mathsf{T}(t_i) K x_\beta \right)^2 + x_\beta K^\mathsf{T} B K x_\beta, \qquad \beta \in \{x, y\} \tag{7.5}$$

with the solution

$$K^\mathsf{T} \left(N^\mathsf{T} W N + B \right) K x_\beta = K^\mathsf{T} N^\mathsf{T} W \mathbf{P}_\beta, \qquad \beta \in \{x, y\}. \tag{7.6}$$

To use this solution in Algorithm 4.1, the weight matrix W just has to be replaced with the iterated weights $Z^{(k)}$.

The original solution $\mathbf{d}_\beta$ afterwards may simply be obtained with

$$\mathbf{d}_\beta = K x_\beta. \tag{7.7}$$

7.2 Enforcing a known direction

To enforce condition (7.3), we first describe $f'(\tau_k - \varepsilon)$ for $\varepsilon > 0$:

$$\begin{aligned} f'(\tau_k - \varepsilon) &\overset{(3.14)}{=} q \sum_{i=1}^{n-1} \frac{d_i - d_{i-1}}{\tau_{i+q} - \tau_i} N_{i,q-1}(\tau_k - \varepsilon) \\ &\overset{3.5(b)}{=} q \sum_{i=k-q}^{k-1} \frac{d_i - d_{i-1}}{\tau_{i+q} - \tau_i} N_{i,q-1}(\tau_k - \varepsilon) \\ &\overset{(7.4)}{=} q \frac{d_k - d_{k-q-1}}{\tau_k - \tau_{k-q}} N_{k-q,q-1}(\tau_k - \varepsilon), \end{aligned}$$

thus

$$f'(\tau_k - \varepsilon) = q \frac{d_k - d_{k-q-1}}{\tau_k - \tau_{k-q}} N_{k-q,q-1}(\tau_k - \varepsilon). \tag{7.8}$$

Similarly, we receive

$$f'(\tau_{k+1} + \varepsilon) = q \frac{d_{k+1} - d_k}{\tau_{k+q+1} - \tau_{k+1}} N_{k+1,q-1}(\tau_{k+1} + \varepsilon). \tag{7.9}$$

Since we have just been interested in $f'(\tau_k - \varepsilon)$ and $f'(\tau_{k+1} + \varepsilon)$ being linearly dependent on δ, we receive additional boundary conditions as

$$\begin{aligned} d_k - d_{k-q-1} &= \mu\delta, \qquad \text{for some } \mu \in \mathbb{R}, \\ d_{k+1} - d_k &= \lambda\delta, \qquad \text{for some } \lambda \in \mathbb{R}. \end{aligned} \tag{7.10}$$

Note that (7.10) actually couples the single dimensions of the approximation problem that we considered separately up to now.

Solving both approximation problems in one step is nevertheless straightforward. As we know that the situation with general ρ might be reduced to successive least squares problems using Algorithm 4.1, for purely notational purposes we will only examine the least squares problem.

By reason of this we just have to take care to use different weights $w_{x,i}$ and $w_{y,i}$ for each dimension because those may take different values in the IRLS algorithm.

Both problems without the boundary conditions are perfectly decoupled. So it is obvious that the sum of the target functions of our approximation problems, which is

$$\begin{aligned}
&\sum_{i=0}^{m-1} w_{x,i}\left(P_{x,i} - N_q^{\mathsf{T}}(t_i)\mathbf{d}_x\right)^2 + \mathbf{d}_x B\mathbf{d}_x + \sum_{i=0}^{m-1} w_{y,i}\left(P_{y,i} - N_q^{\mathsf{T}}(t_i)\mathbf{d}_y\right)^2 + \mathbf{d}_y B\mathbf{d}_y \\
&\quad = \left\| \sqrt{W_x}(\mathbf{P}_x - N\mathbf{d}_x)\right\|_2^2 + \left\| \sqrt{W_y}(\mathbf{P}_y - N\mathbf{d}_y)\right\|_2^2 + \begin{bmatrix}\mathbf{d}_x & \mathbf{d}_y\end{bmatrix}\begin{bmatrix}B & 0\\ 0 & B\end{bmatrix}\begin{bmatrix}\mathbf{d}_x\\ \mathbf{d}_y\end{bmatrix} \\
&\quad = \left\| \begin{bmatrix}\sqrt{W_x} & 0\\ 0 & \sqrt{W_y}\end{bmatrix}\left(\begin{bmatrix}P_x\\ P_y\end{bmatrix} - \begin{bmatrix}N & 0\\ 0 & N\end{bmatrix}\begin{bmatrix}\mathbf{d}_x\\ \mathbf{d}_y\end{bmatrix}\right)\right\|_2^2 + \begin{bmatrix}\mathbf{d}_x & \mathbf{d}_y\end{bmatrix}\begin{bmatrix}B & 0\\ 0 & B\end{bmatrix}\begin{bmatrix}\mathbf{d}_x\\ \mathbf{d}_y\end{bmatrix}
\end{aligned}$$

has the same minimum solution as the separate problems. Writing

$$N_{xy} := \begin{bmatrix}N & 0\\ 0 & N\end{bmatrix} \qquad W_{xy} := \begin{bmatrix}W_x & 0\\ 0 & W_y\end{bmatrix} \qquad B_{xy} := \begin{bmatrix}B & 0\\ 0 & B\end{bmatrix}$$

we gain a new linear least squares problem

$$\min_{[d_x,d_y]^{\mathsf{T}}}\left(\left\| \sqrt{W_{xy}}\left(\begin{bmatrix}P_x\\ P_y\end{bmatrix} - N_{xy}\begin{bmatrix}\mathbf{d}_x\\ \mathbf{d}_y\end{bmatrix}\right)\right\|_2^2 + \begin{bmatrix}\mathbf{d}_x & \mathbf{d}_y\end{bmatrix} B_{xy}\begin{bmatrix}\mathbf{d}_x\\ \mathbf{d}_y\end{bmatrix}\right). \tag{7.11}$$

The boundary conditions (7.4) and (7.10) again are linear and may be treated the same way as was done in chapter 7.1. By including the two additional variables μ and λ it is easy to see that those boundary conditions can be written as

$$\underbrace{\begin{bmatrix}0_{q+2,k-q-2} & \overline{E} & 0_{q+2,n-q-3} & 0_{q+2,q+3} & 0_{q+2,n-k-1} & \Delta_x\\ 0_{q+2,k-q-2} & 0_{q+2,q+3} & 0_{q+2,n-q-3} & \overline{E} & 0_{q+2,n-k-1} & \Delta_y\end{bmatrix}}_{\in\mathbb{R}^{2q+4\times 2n+2}}\begin{bmatrix}d_x\\ d_y\\ \mu\\ \lambda\end{bmatrix} = 0$$

with

$$\overline{E} := \begin{bmatrix}1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1\end{bmatrix} \in \mathbb{R}^{q+2\times q+3}$$

and

$$\Delta_\beta := \begin{bmatrix}\delta_\beta & 0\\ 0 & 0\\ \vdots & \vdots\\ 0 & 0\\ 0 & \delta_\beta\end{bmatrix} \in \mathbb{R}^{q+2\times 2}, \qquad \beta \in \{x, y\}.$$

A basis of the kernel of this boundary condition is

$$K := \left[\begin{array}{cccccc|c} 1_{k-q-2} & & & & & & 0_{k-q-2,2}\\ & \overline{K} & & & & & K_{\Delta_x}\\ & & 1_{n-q-3} & & & & 0_{n-q-3,2}\\ & & & \overline{K} & & & K_{\Delta_y}\\ & & & & & 1_{n-k-1} & 0_{n-k-1,2}\\ \hline & & & 0_{2,2(n-q-2)} & & & 1_2 \end{array}\right] \in \mathbb{R}^{2n+2\times 2(n-q-1)}$$

with

$$\overline{K} := \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{q+3}$$

and

$$K_{\Delta_\beta} := \begin{bmatrix} -\delta_\beta & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & \delta_\beta \end{bmatrix} \in \mathbb{R}^{q+3\times 2}.$$

To keep the matrix K well conditioned it is advisable to choose a δ such that $\|\delta\|_2 = 1$.

The optimization problem now has to be altered insofar that it also includes the free variables μ and λ:

$$\min_{[d_x,d_y,\mu,\lambda]^\mathsf{T}} \left(\left\| \sqrt{W_{xy}} \left(\begin{bmatrix} P_x \\ P_y \end{bmatrix} - \begin{bmatrix} N_{xy} & 0_{2m,2} \end{bmatrix} \begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mu \\ \lambda \end{bmatrix} \right) \right\|_2^2 + \begin{bmatrix} \mathbf{d}_x & \mathbf{d}_y & \mu & \lambda \end{bmatrix} \begin{bmatrix} B_{xy} & \\ & 0_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mu \\ \lambda \end{bmatrix} \right).$$

Replacing the variable vector and solving the linear system similarly to (6.10) we get

$$K^\mathsf{T} \begin{bmatrix} \left(N_{xy}^\mathsf{T} W_{xy} N_{xy} + B_{xy}\right) & 0_{2m,2} \\ 0_{2,2m} & 0_{2,2} \end{bmatrix} Kx = K^\mathsf{T} \begin{bmatrix} N_{xy}^\mathsf{T} \\ 0_{2,2m} \end{bmatrix} W_{xy} \begin{bmatrix} P_x \\ P_y \end{bmatrix} \tag{7.12}$$

and

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mu \\ \lambda \end{bmatrix} = Kx. \tag{7.13}$$

If the standstill is at the beginning or end of the recorded data this process has to be adapted slightly. We will only need one boundary condition for each dimension. Thus we will need to delete two rows from the boundary condition matrix. Furthermore, only one additional free variable will be necessary. Those modifications result in a marginally different kernel basis. As the same general idea is used and the result looks mostly identical, we will omit the detailed description of the process.

7.3 Enforcing a constant, unknown direction

Finally we have to discuss how to handle (7.2) where a constant, yet unknown direction shall be enforced. According to equations (7.8) and (7.9) the correct condition can easily be given as linear dependence of the difference of control points:

$$\lambda(d_k - d_{k-q-1}) + \mu(d_{k+1} - d_k) = 0. \tag{7.14}$$

Again the x- and y-dimension are coupled by the boundary condition. In contrast to the previous chapter the free parameters μ and λ now are *multiplied* by control points and not constants. Thus, we yield a quadratic boundary condition. This aspect will result in a significantly more difficult approximation problem.

The matrix containing the boundary conditions now becomes a matrix valued function of μ and λ. We do not need to combine the dimensions into one single approximation problem, though. The boundary conditions are

$$\begin{bmatrix} 0_{q+1,k-q-2} & \overline{E}(\mu,\lambda) & 0_{q+1,n-k-1} \end{bmatrix} \mathbf{d}_\beta = 0, \qquad \beta \in \{x, y\}$$

with

$$\overline{E}(\mu,\lambda) := \begin{bmatrix} -\lambda & & & & \lambda-\mu & \mu \\ & 1 & -1 & & & \\ & & \ddots & \ddots & & \\ & & & 1 & -1 & \end{bmatrix} \in \mathbb{R}^{q+1 \times q+3}.$$

The basis of the kernel, which means a basis for the valid solutions, also may be formulated in a similar fashion as above:

$$K(\mu,\lambda) := \begin{bmatrix} 1_{k-q-2} & & \\ & \overline{K}(\mu,\lambda) & \\ & & 1_{n-k-1} \end{bmatrix} \in \mathbb{R}^{n \times n-q-1}, \tag{7.15}$$

with

$$\overline{K}(\mu,\lambda) := \begin{bmatrix} 1 & \mu \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 1 & \lambda \end{bmatrix} \in \mathbb{R}^{q+3 \times 2}.$$

Letting both μ and λ vary, introduces an additional degree of freedom that technically is not necessary and was just included for symmetry reasons. In terms of numerical stability we will not get problems as long as both μ and λ stay within a reasonable magnitude, i.e. neither μ nor λ go close to infinity or zero. In the following we will therefore keep the parameter λ fixed and write

$$K(\mu,\lambda) =: K_\lambda(\mu), \qquad \lambda \in \mathbb{R}. \tag{7.16}$$

Substituting $\mathbf{d}_x = K_\lambda(\mu)x_x$ and $\mathbf{d}_y = K_\lambda(\mu)x_y$ this yields the constrained robust approximation problem

$$\min_{[x_x,x_y,\mu]} \sum_{\beta\in\{x,y\}} \left(\sum_{i=0}^{m-1} w_{\beta,i} \rho\Big(P_{\beta,i} - N_q^\mathsf{T}(t_i) K_\lambda(\mu) x_\beta \Big) + x_\beta^\mathsf{T} K_\lambda^\mathsf{T}(\mu) B K_\lambda(\mu) x_\beta \right).$$

This is a non-linear approximation problem as discussed in chapter 4.2. Thus the derived theory for robust regression problems may be applied and the problem may be reduced to a series of least squares problems. In vector notation we get

$$\min_{x_x,x_y,\mu} \sum_{\beta\in\{x,y\}} \left(\left\| \sqrt{W_\beta} \Big(P_\beta - N K_\lambda(\mu) x_\beta \Big) \right\|_2^2 + x_\beta^\mathsf{T} K_\lambda^\mathsf{T}(\mu) B K_\lambda(\mu) x_\beta \right). \tag{7.17}$$

The least squares part of this problem can be easily seen to be separable (compare to equation (4.9)). Using the Cholesky-like decomposition C such that $B = C^\mathsf{T}C$, introduced in equation (6.8), we note that

$$\begin{aligned}
&\left\| \sqrt{W_\beta}\left(P_\beta - NK_\lambda(\mu)x_\beta\right)\right\|_2^2 + x_\beta^\mathsf{T}K_\lambda^\mathsf{T}(\mu)BK_\lambda(\mu)x_\beta \\
&\quad = \left\| \sqrt{W_\beta}P_\beta - \sqrt{W_\beta}NK_\lambda(\mu)x_\beta\right\|_2^2 + \left\|CK_\lambda(\mu)x_\beta\right\|_2^2 \\
&\quad = \left\| \begin{bmatrix} \sqrt{W_\beta}P_\beta \\ 0_{n,1} \end{bmatrix} - \begin{bmatrix} \sqrt{W_\beta}N \\ C \end{bmatrix} K_\lambda(\mu)x_\beta \right\|_2^2.
\end{aligned}$$

Now (7.17) becomes

$$\min_{x_x,x_y,\mu} \sum_{\beta\in\{x,y\}} \left\| \begin{bmatrix} \sqrt{W_\beta}P_\beta \\ 0_{n,1} \end{bmatrix} - \begin{bmatrix} \sqrt{W_\beta}N \\ C \end{bmatrix} K_\lambda(\mu)x_\beta \right\|_2^2.$$

This problem is separable and, according to Theorem 4.12, equivalent to solving

$$\min_{\mu} \min_{x_x,x_y} \sum_{\beta\in\{x,y\}} \left\| \begin{bmatrix} \sqrt{W_\beta}P_\beta \\ 0_{n,1} \end{bmatrix} - \begin{bmatrix} \sqrt{W_\beta}N \\ C \end{bmatrix} K_\lambda(\mu)x_\beta \right\|_2^2,$$

because $K(\mu)$ has full rank for all μ as long as $\lambda \neq 0$.

Using the fact that both dimensions are only coupled by μ the optimization problem is identical to

$$\min_{\mu} \left(g_x(\mu) + g_y(\mu)\right) \tag{7.18}$$

with

$$g_\beta(\mu) = \min_{x_\beta} \left\| \begin{bmatrix} \sqrt{W_\beta}P_\beta \\ 0_{n,1} \end{bmatrix} - \begin{bmatrix} \sqrt{W_\beta}N \\ C \end{bmatrix} K_\lambda(\mu)x_\beta \right\|_2^2, \qquad \beta \in \{x,y\}. \tag{7.19}$$

To find an optimal value for (7.18) we are able to give an explicit expression for the functions $g_\beta(\mu)$.

Theorem 7.20. *Let $\lambda \neq 0$, $A = [a_0, \ldots, a_{n-1}] \in \mathbb{R}^{m\times n}$ for $m \in \mathbb{N}$, $y \in \mathbb{R}^m$, $n' := n - q - 1$, $P \in \mathbb{R}^{n'\times n'}$ a permutation matrix that exchanges the $(k-q)$-th column of a matrix with the n'-th column and $K_\mu(\lambda)$ as in equation* (7.15)*. Furthermore let $Q_\lambda R_\lambda$ be QR-decomposition such that*

$$AK_\lambda(0)P = Q_\lambda R_\lambda = \begin{bmatrix} Q_{\lambda,0} & Q_{\lambda,1} \end{bmatrix} \begin{bmatrix} R_{\lambda,0} \\ 0 \end{bmatrix},$$

where $R_{\lambda,0}$ is upper triangular, Q_λ orthogonal, z_λ the last column of $Q_{\lambda,0}$ and r_λ the bottom right entry of $R_{\lambda,0}$. Then

$$g(\mu) := \min_x \|y - AK_\lambda(\mu)x\|_2^2 = \frac{((\theta_\lambda\eta_\lambda - \gamma_\lambda\zeta_\lambda)\mu + \eta_\lambda r_\lambda)^2}{\gamma((\theta_\lambda\mu + r_\lambda)^2 + \gamma_\lambda\mu^2)} + c$$

with

$$\theta_\lambda = z_\lambda^\mathsf{T}a_{k-q-2}, \qquad \eta_\lambda = a_{k-q-2}^\mathsf{T}Q_{\lambda,1}Q_{\lambda,1}^\mathsf{T}y, \qquad \gamma_\lambda = a_{k-q-2}^\mathsf{T}Q_{\lambda,1}Q_{\lambda,1}^\mathsf{T}a_{k-q-2}, \qquad \zeta_\lambda = z_\lambda^\mathsf{T}y$$

and some $c \in \mathbb{R}$.

Convention 7.21. For brevity we will omit the fixed parameter λ in the following and just write $K(\mu)$, Q, R, Q_0, Q_1, R_0, z and r as well as θ, η, γ and ζ.

The proof is conceptually simple while at the same time technically laborious. We will use the variable projection approach of Golub and Pereyra. The important idea is to explicitly find the QR-decomposition of $AK(\mu)P$ as a function of μ. Note that the permutation matrix P just pivots the columns of $AK(\mu)$ in a way that finding this decomposition becomes possible.

Lemma 7.22. *Let the notation be as in Theorem 7.20. The matrix*

$$\tilde{R} := G(\mu)\tilde{Q}Q^{\mathsf{T}}AK(\mu)P$$

with

$$G(\mu) := \begin{bmatrix} 1_{n'-1} & & & \\ & \frac{r+\theta\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & \frac{\sqrt{\gamma}\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & \\ & -\frac{\sqrt{\gamma}\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & \frac{r+\theta\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & \\ & & & 1_{m-n'-1} \end{bmatrix}$$

and

$$\tilde{Q} := \begin{bmatrix} 1_{n'} & \\ & 1_{m-n'} - 2vv^{\mathsf{T}} \end{bmatrix}, \qquad v := \frac{Q_1^{\mathsf{T}}a_{k-q-2} - \sqrt{\gamma}e_0}{\|Q_1^{\mathsf{T}}a_{k-q-2} - \sqrt{\gamma}e_0\|_2}$$

is upper triangular and $G(\mu)$ and $\tilde{Q}$ are orthogonal, i.e. $Q\tilde{Q}^{\mathsf{T}}G^{\mathsf{T}}(\mu)\tilde{R}$ is a QR-decomposition of $AK(\mu)P$.

The idea of the transformation that yields the QR decomposition is schematically displayed in Figure 7.1.

Proof. The proof is divided in two parts: First, we will show that $\tilde{R}$ in fact is upper triangular. Only afterwards we verify that the given matrices form a QR decomposition of $AK(\mu)P$.

First note that $\gamma = \left\|Q_1^{\mathsf{T}}a_{k-q-2}\right\|_2$. The matrix $1_{m-n'} - 2vv^{\mathsf{T}}$ is a Householder transformation that reflects the vector $Q_1^{\mathsf{T}}a_{k-q-2}$ on the vector $\sqrt{\gamma}e_0$ and is orthogonal [36]. Thus we have

$$\begin{aligned}\tilde{Q}(R + \mu Q^{\mathsf{T}}a_{k-q-2}e_{n'-1}^{T}) &= \begin{bmatrix} 1_{n'} & \\ & 1_{m-n'} - 2vv^{\mathsf{T}} \end{bmatrix}\left(\begin{bmatrix} R_0 \\ 0 \end{bmatrix} + \mu \begin{bmatrix} Q_0^{\mathsf{T}}a_{k-q-2} \\ Q_1^{\mathsf{T}}a_{k-q-2} \end{bmatrix} e_{n'-1}^{\mathsf{T}}\right) \\ &= \begin{bmatrix} R_0 \\ 0 \end{bmatrix} + \mu \begin{bmatrix} Q_0^{\mathsf{T}}a_{k-q-2} \\ \sqrt{\gamma}e_0 \end{bmatrix} e_{n'-1}^{\mathsf{T}}.\end{aligned}$$

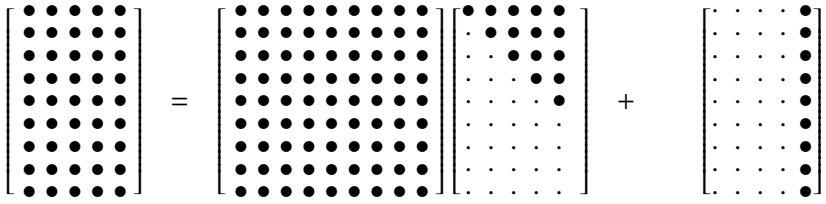
This matrix is close to upper triangular with only the entry at index $n', n'-1$ disturbing the pattern and having the value $\mu\sqrt{\gamma}$. The matrix entry at index $n'-1, n'-1$ equals

$$r + \theta\mu.$$

Thus the matrix $G(\mu)$ is a Givens rotation matrix [36] that removes the value at position $n', n'-1$. Furthermore $G(\mu)$ is orthogonal. It follows that

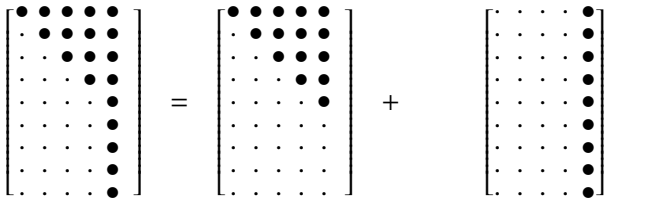
$$\tilde{R} = G(\mu)\tilde{Q}(R + \mu Q^{\mathsf{T}}a_{k-q-2}e_{n'-1}^{\mathsf{T}})$$

is upper triangular.



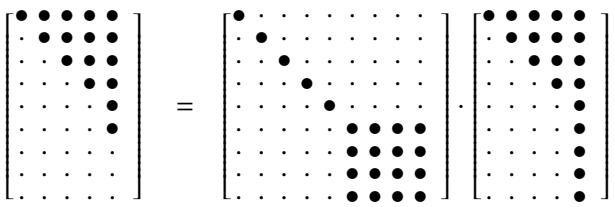
$AK(\mu)P$ $\qquad$ $Q \cdot R$ $\qquad$ $A(K(\mu) - K(0))P$

(a) To find a QR decomposition of $AK(\mu)P$ we have the decomposition of $AK(0)P$ available.



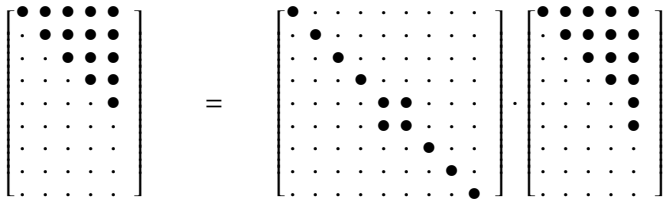
$Q^\mathsf{T}AK(\mu)P$ $\qquad$ R $\qquad$ $Q^\mathsf{T}A(K(\mu) - K(0))P$

(b) By multiplying (a) with Q^T we already have a matrix that is close to upper triangular.



$\tilde{Q}Q^\mathsf{T}AK(\mu)P$ $\qquad$ $\tilde{Q}$ $\qquad$ $Q^\mathsf{T}AK(\mu)P$

(c) The Householder transformation $\tilde{Q}$ removes most non-zero elements in the last column without altering the remaining zero elements.



$G(\mu)\tilde{Q}Q^\mathsf{T}AK(\mu)P$ $\qquad$ $G(\mu)$ $\qquad$ $\tilde{Q}Q^\mathsf{T}AK(\mu)P$

(d) After one final Givens rotation $G(\mu)$ the matrix product has upper triangular form. The matrix Q of the QR decomposition now simply can be obtained by collecting the inverse of the transformation matrices used: $Q\tilde{Q}^\mathsf{T}G^\mathsf{T}(\mu)$.

Figure 7.1: Schematic display of the effects of various matrix operations used in the process of finding a QR decomposition of $AK(\mu)P$. Possible sparsity of A is not taken into consideration.

Finally we observe that

$$A\left(K(\mu) - K(0)\right) = \mu a_{k-q-2} e_{k-q-1}^{\mathsf{T}}.$$

Using $QQ^{\mathsf{T}} = 1_m$ we get

$$\begin{aligned}
\tilde{R} &= G(\mu)\tilde{Q}(R + \mu Q^{\mathsf{T}} a_{k-q-2} e_{n'-1}^{\mathsf{T}}) \\
&= G(\mu)\tilde{Q}Q^{\mathsf{T}}(QRP^{\mathsf{T}} + \mu a_{k-q-2} e_{n'-1}^{\mathsf{T}} P^{\mathsf{T}})P \\
&= G(\mu)\tilde{Q}Q^{\mathsf{T}}(AK(0)PP^{\mathsf{T}} + \mu a_{k-q-2} e_{k-q-1}^{\mathsf{T}})P \\
&= G(\mu)\tilde{Q}Q^{\mathsf{T}}\left(AK(0) + A\left(K(\mu) - K(0)\right)\right)P \\
&= G(\mu)\tilde{Q}Q^{\mathsf{T}} AK(\mu)P
\end{aligned}$$

and hence

$$AK(\mu)P = (Q\tilde{Q}^{\mathsf{T}} G^{\mathsf{T}}(\mu))\tilde{R}. \qquad \square$$

Note that the previous proof especially reveals

$$\tilde{R} = G(\mu)\tilde{Q}(R + \mu Q^{\mathsf{T}} a_{k-q-2} e_{n'-1}^{\mathsf{T}}) = G(\mu)\left(R + \mu \begin{bmatrix} Q_0^{\mathsf{T}} a_{k-q-2} \\ \sqrt{\gamma} e_0 \end{bmatrix} e_{n'-1}^{\mathsf{T}}\right). \tag{7.23}$$

Proof of Theorem 7.20. $g(\mu)$ is a separable problem, thus we may apply the variable projection approach of Golub and Pereyra, in particular the reformulation given in equation (4.13). According to Lemma 7.22 we have the orthogonal matrix Q of the QR-decomposition of $AK(\mu)$ given as

$$Q\tilde{Q}^{\mathsf{T}} G^{\mathsf{T}}(\mu).$$

The last $m - n'$ columns which are needed to use the representation as in equation (4.13) may be selected by removing the first n' rows of $G(\mu)$. This yields the identity

$$g(\mu) = \left\| \begin{bmatrix} 0_{1,n'-1} & -\frac{\sqrt{\gamma}\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & \frac{r+\theta\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & 0_{1,m-n'-1} \\ 0_{m-n'-1,n'-1} & 0_{m-n'-1,1} & 0_{m-n'-1,1} & 1_{m-n'-1} \end{bmatrix} \tilde{Q}Q^{\mathsf{T}} y \right\|_2^2.$$

Only the first entry of the vector within the norm above depends on μ. Thus all other entries will be constant and there exists some $c \in \mathbb{R}$ such that

$$g(\mu) = \left\| \begin{bmatrix} 0_{1,n'-1} & -\frac{\sqrt{\gamma}\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & \frac{r+\theta\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & 0_{m-n'-1,1} \end{bmatrix} \tilde{Q}Q^{\mathsf{T}} y \right\|_2^2 + c.$$

For the following computation we are only interested in the values of $\tilde{Q}Q^{\mathsf{T}} y$ that have the indices $n' - 1$ and n'. As we remember

$$\tilde{Q} = \begin{bmatrix} 1_{n'} & \\ & 1_{m-n'} - 2vv^{\mathsf{T}} \end{bmatrix},$$

the first of those entries just is the value at index $n' - 1$ of $Q^{\mathsf{T}} y$ which is $z^{\mathsf{T}} y = \zeta$. The second entry requires a bit more effort. With $Q = [Q_0, Q_1]$, $Q_0 \in \mathbb{R}^{n' \times n'}$ and $v = [v_0, \ldots, v_{m-n'-1}]$ the required value equals

$$(e_0^{\mathsf{T}} - 2v_0 v^{\mathsf{T}})Q_1^{\mathsf{T}} y.$$

Let $x := Q_1^\mathsf{T} a_{k-q-2} =: [x_0, \ldots, x_{m-n'-1}]$. According to the definition of γ in Theorem 7.20 we have $\|x\|_2^2 = \gamma$ and

$$v = \frac{x - \sqrt{\gamma} e_0}{\|x - \sqrt{\gamma} e_0\|_2}.$$

Furthermore

$$\begin{aligned} 2v_0 v^\mathsf{T} &= \frac{2(x_0 - \sqrt{\gamma})(x^\mathsf{T} - \sqrt{\gamma} e_0^\mathsf{T})}{\|x - \sqrt{\gamma} e_0\|_2^2} = \frac{2(x_0 - \sqrt{\gamma})(x^\mathsf{T} - \sqrt{\gamma} e_0^\mathsf{T})}{\gamma - x_0^2 + (x_0 - \sqrt{\gamma})^2} \\ &= \frac{2(x_0 - \sqrt{\gamma})(x^\mathsf{T} - \sqrt{\gamma} e_0^\mathsf{T})}{2(\gamma - x_0\sqrt{\gamma})} = -\frac{x^\mathsf{T}}{\sqrt{\gamma}} + e_0^\mathsf{T}. \end{aligned}$$

It follows that

$$(e_0^\mathsf{T} - 2v_0 v^\mathsf{T}) Q_1^\mathsf{T} y = \frac{x^\mathsf{T}}{\sqrt{\gamma}} Q_1^\mathsf{T} y = \frac{\eta}{\sqrt{\gamma}}.$$

Altogether we have

$$\begin{aligned} g(\mu) &= \left\| \begin{bmatrix} -\frac{\sqrt{\gamma}\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} & \frac{r+\theta\mu}{\sqrt{(r+\theta\mu)^2+\gamma\mu^2}} \end{bmatrix} \begin{bmatrix} \zeta \\ \frac{\eta}{\sqrt{\gamma}} \end{bmatrix} \right\|_2^2 + c \\ &= \frac{\left(\frac{\eta(r+\theta\mu)}{\sqrt{\gamma}} - \zeta\sqrt{\gamma}\mu\right)^2}{(r+\theta\mu)^2 + \gamma\mu^2} + c = \frac{((\theta\eta - \gamma\zeta)\mu + \eta r)^2}{\gamma\left((r+\theta\mu)^2 + \gamma\mu^2\right)} + c. \end{aligned}$$

□

In total we have shown that the functions g_β are rational functions

$$g_\beta(\mu) = \frac{((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)^2}{\gamma_\beta((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2)} + c_\beta, \qquad \beta \in \{x, y\} \tag{7.24}$$

with the coefficients $\theta_\beta, \gamma_\beta, \zeta_\beta, \eta_\beta$ and r_β defined as in Theorem 7.20. This brings us a huge step closer towards the solution of the optimization problem

$$\min_\mu g(\mu) := \min_\mu \left(g_x(\mu) + g_y(\mu)\right).$$

Finding a minimum now may be achieved by finding all zeros of the derivative $g'(\mu)$ and selecting the zero with the smallest value of $g(\mu)$. First, we note that

$$g'(\mu) = g_x'(\mu) + g_y'(\mu).$$

Lemma 7.25.

$$g_\beta'(\mu) = \frac{-2r_\beta((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)((\zeta_\beta\theta_\beta + \eta_\beta)\mu + \zeta_\beta r_\beta)}{\left((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2\right)^2}, \qquad \beta \in \{x, y\}.$$

Proof. The result follows from the quotient rule and straightforward simplifications:

$$\begin{aligned}
g'_\beta(\mu) =& \frac{2(\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2)}{\gamma_\beta((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2)^2} \\
&- \frac{2((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)^2(\theta_\beta(\theta_\beta\mu + r_\beta) + \gamma_\beta\mu)}{\gamma_\beta((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2)^2} \\
=& \frac{2((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)}{\gamma_\beta((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2)^2}\Big((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2) \\
&- ((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)(\theta_\beta(\theta_\beta\mu + r_\beta) + \gamma_\beta\mu)\Big)
\end{aligned}$$

and

$$\begin{aligned}
&(\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)((\theta_\beta\mu + r_\beta)^2 + \gamma_\beta\mu^2) - ((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)(\theta_\beta(\theta_\beta\mu + r_\beta) + \gamma_\beta\mu) \\
&= (\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)(\theta_\beta\mu + r_\beta)^2 - ((\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)\mu + \eta_\beta r_\beta)\theta(\theta_\beta\mu + r_\beta) - \eta_\beta r_\beta\gamma_\beta\mu \\
&= (\theta_\beta\eta_\beta - \gamma_\beta\zeta_\beta)(\theta_\beta\mu + r_\beta)r_\beta - \eta_\beta r_\beta\theta(\theta_\beta\mu + r_\beta) - \eta_\beta r_\beta\gamma_\beta\mu \\
&= -r_\beta((\zeta_\beta\theta_\beta + \eta_\beta)\mu + \zeta_\beta r_\beta)).
\end{aligned}$$

□

Furthermore, the maximal and minimal value g_β may be easily seen:

Corollary 7.26.

$$\arg\min_\mu g_\beta(\mu) = \frac{\eta_\beta r_\beta}{\gamma_\beta\zeta_\beta - \theta_\beta\eta_\beta} \qquad \textit{and} \qquad \arg\max_\mu g_\beta(\mu) = -\frac{\zeta_\beta r_\beta}{\zeta_\beta\theta_\beta + \eta_\beta}$$

for $\beta \in \{x, y\}$.

Proof. Obviously $g'_\beta(\mu)$ has two zeros at the locations above. Looking at $g_\beta(\mu)$ itself one may clearly identify the minimum. Therefore, the second zero has to form a maximum. Moreover we notice by standard theory of rational functions that

$$\lim_{\mu\to\pm\infty} g(\mu) = \frac{(\theta_l\eta_l - \gamma_l\zeta_l)^2}{\gamma_l^2 + \theta_l^2}. \tag{7.27}$$

As this limit is constant, the minimum and maximum above in fact are global optimal values. □

Corollary 7.28.

$$\lim_{\mu\to\pm\infty} g(\mu) > \inf_\mu g(\mu).$$

Proof. From equation (7.27) it follows that

$$\lim_{\mu\to-\infty} g(\mu) = \lim_{\mu\to\infty} g(\mu).$$

The asymptotic behaviour of $g'(\mu)$ is identical to the behaviour of $c\mu^{-2}$, hence

$$\operatorname{sign}(g'(-\mu)) = \operatorname{sign}(g'(\mu))$$

for large enough μ. This shows that either the value of $g(\mu)$ is decreasing for small μ or increasing for large μ and especially that there are μ with function values smaller than the limiting value. □

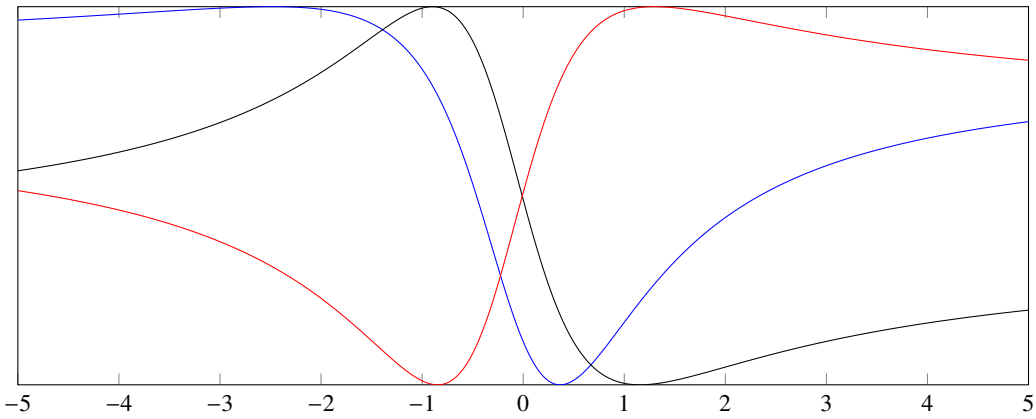


Figure 7.2: Three typical graphs of $g(\mu)$ (with the image of $g(\mu)$ normalized) as they arise in practical approximation situations. The red line comes from a pushback process.

Due to Corollary 7.28 we know that the minimal value of $g(\mu)$ has to be located at a zero of $g'(\mu)$. This function is, as a sum of rational functions, again rational. Most importantly the numerator is a polynomial of degree six and therefore might have as much as six real zeros. Empirical results have shown that in most cases there are only two zeros though.

Without loss of generality we will assume that g_x and g_y have different minimal values, i.e.

$$\frac{\eta_x r_x}{\gamma_x \zeta_x - \theta_x \eta_x} \neq \frac{\eta_y r_y}{\gamma_y \zeta_y - \theta_y \eta_y}. \tag{7.29}$$

Otherwise the minimum of $g(\mu)$ clearly is taken at $\frac{\eta_x r_x}{\gamma_x \zeta_x - \theta_x \eta_x}$.

The general idea we will follow is to divide $\mathbb{R}$ into intervals that guarantee g_x and g_y both being monotone within. Therefore, let $a_1, \ldots, a_4$ be the zeros of g_x' and g_y' ordered so that

$$a_1 \leq a_2 \leq a_3 \leq a_4.$$

Then

$$\mathbb{R} = (-\infty, a_1] \cup (a_1, a_2] \cup (a_2, a_3] \cup (a_3, a_4] \cup (a_4, \infty).$$

We will denote $a_0 := -\infty$, $a_5 = \infty$. In the following we will only examine the interior of the intervals above. The only case the minimum may lie on the boundary of one intervals has been handled in equation (7.29).

Obviously, the requirement

$$g'(\mu) = g_x'(\mu) + g_y'(\mu) = 0$$

only may be fulfilled if g_x' and g_y' have different signs. As by construction the sign of g_x' and g_y' does not change within one interval we immediately get a first condition on the existence of a minimum:

$$\arg\min_{\mu} g(\mu) \notin (a_i, a_{i+1}) \quad \text{if} \quad g_x'\left(\frac{a_i + a_{i+1}}{2}\right) g_y'\left(\frac{a_i + a_{i+1}}{2}\right) \geq 0. \tag{7.30}$$

Using this observation out of the five intervals possibly containing the minimum we are able to rule out at least two, as the sign of the product of the derivatives has to be alternating between the different intervals.

We may easily obtain the numerator polynomial $h(\mu)$ of $g(\mu)$ numerically. A corresponding symbolic computation of course also is possible, but technically exhausting. In the following we will therefore state how the minimum may be found without conducting the computations itself. A reference on the numerical methods may be found in Appendix A.2.

Now all that we have to do is to find all zeros of the polynomial $h(\mu)$. The amount of algorithms to locate zeros of polynomials is so immense that there are entire books written on that single, but important topic [52], [53]. As the degree of h is rather low, we will omit the introduction of overly complicated methods and follow a simple approach based on Newton's method and Sturm's theorem.

We obtain the numerator polynomial

$$h(\mu) = \sum_{i=0}^{6} h_i \mu^i.$$

The Gershgorin circle theorem [6] guarantees that the absolute values of all zeros of h are less than

$$\frac{\max\{|h_0|, 1 + |h_1|, \ldots, 1 + |h_5|\}}{|h_6|}.$$

Hence we may set

$$a_0 := -\frac{\max\{|h_0|, 1 + |h_1|, \ldots, 1 + |h_5|\}}{|h_6|}, \qquad a_5 := \frac{\max\{|h_0|, 1 + |h_1|, \ldots, 1 + |h_5|\}}{|h_6|}.$$

Now let us recall Sturm's theorem: For a polynomial p let the Sturm polynomials be defined as

$$\begin{aligned} p_0(x) &:= p(x), \\ p_1(x) &:= p'(x), \\ p_i(x) &:= -\operatorname{rem}(p_{i-2}(x), p_{i-1}(x)), \qquad i \geq 2. \end{aligned}$$

Here $\operatorname{rem}(p_{i-2}(x), p_{i-1}(x))$ denotes the remainder of the polynomial division of p_{i-2} by p_{i-1}. Those polynomials can be computed by polynomial long division.

Theorem 7.31 (Sturm, 1829 [14, p. 44])**.** *Let $\sigma(\xi)$ equal the number of sign changes in the sequence $p_0(\xi), \ldots, p_m(\xi)$ where m is chosen such that $p_{m+1} \equiv 0$. Then $\sigma(a) - \sigma(b)$ equals the number of distinct real roots of p in the interval $(a, b]$.*

Note that if $m < 6$ the polynomial has multiple roots. We never observed this case in practical applications, however if it may happen we can remove the multiple root using Euclid's algorithm.

Using Sturm's theorem we can count how many roots are to be expected within each of the intervals (a_i, a_{i+1}). If we just have a single root within the interval we may use a combination of the binary search and Newton's method (see Appendix A.3) to identify the exact location of the root.

If, on the other hand, we have multiple roots within one interval, we just take the half of the interval and employ Sturm's theorem again. This method is not the most efficient possible but

as in practical situations each interval usually contains one root, we do not need to use a more sophisticated algorithm.

In total we gain:

Algorithm 7.1: Root finding of $h(\mu)$

Input: Polynomial function $h(\mu)$.
Breaks $a_0, \ldots, a_5$.
Result: Set $\mathcal{M}$ containing all real zeros of h.

$p_0, \ldots, p_m \leftarrow$ Sturm polynomials of $h(x)$
$\sigma(\mu) \leftarrow$ Function that computes the sign changes of the Sturm polynomials
for $i \leftarrow 0$**to** 4 **do**
  $\mathcal{M} \leftarrow \mathcal{M} \cup$ findZeros$(h, \sigma, a_i, a_{i+1})$
end for

procedure findZeros(h, σ, a, b, s, s')
  if $\sigma(a) - \sigma(b) = 0$ **then**
    return $\emptyset$
  else if $\sigma(a) - \sigma(b) = 1$ **then**
    return {newtonBisectionHybrid(h, a, b)}
  else
    return findZeros$(h, \sigma, a, \frac{a+b}{2}) \cup$ findZeros$(h, \sigma, \frac{a+b}{2}, b)$
  end if
end procedure

After using this algorithm we get $\mu_0, \ldots, \mu_{j-1}$ such that

$$g'(\mu_i) = h(\mu_i) = 0, i = 0, \ldots, j-1$$

and select

$$\hat{\mu} := \arg \min_{\{\mu_0 \cdots, \mu_{j-1}\}} g(\mu).$$

Now the very last step is to use the optimal value $\hat{\mu}$ to gain the solution to the whole spline fitting problem (7.17), namely the values for x_x and x_y or, substituting back, the original spline control points $\mathbf{d}_x$ and $\mathbf{d}_y$. Again we may solve the decoupled problems separately to find x_β that solves the optimization problem in $g_\beta(\hat{\mu})$ from equation (7.19). Using the same notation as in Theorem 7.20 this equals the problem

$$\min_x \|y - AK(\hat{\mu})x\|_2^2. \tag{7.32}$$

According to Lemma 7.22 we have a QR-decomposition

$$AK(\hat{\mu}) = Q\tilde{Q}^\mathsf{T} G^\mathsf{T}(\hat{\mu})\tilde{R}P^T.$$

This gives us the equivalent approximation problem

$$\min_x \left\| y - Q\tilde{Q}^\mathsf{T} G^\mathsf{T}(\hat{\mu})\tilde{R}P^T x \right\|_2^2.$$

We know that the solution to this problem can be found with (4.7)

$$P\tilde{R}^{\mathsf{T}}G(\hat{\mu})\tilde{Q}Q^{T}Q\tilde{Q}^{T}G^{T}(\hat{\mu})\tilde{R}P^{\mathsf{T}}x = P\tilde{R}^{\mathsf{T}}G(\hat{\mu})\tilde{Q}Q^{T}y$$

which is equivalent to

$$\tilde{R}^{\mathsf{T}}\left(\tilde{R}P^{\mathsf{T}}x - G(\hat{\mu})\tilde{Q}Q^{T}y\right) = 0.$$

Beyond that

$$\tilde{R} = \begin{bmatrix}\tilde{R}_1\\ 0\end{bmatrix},$$

where $\tilde{R}_1$ is upper triangular and of full rank. So we may remove additional zeros out of this sum, multiply by $\tilde{R}_1^{-1}$ and yield yet another expression for the solution of (7.32):

$$\tilde{R}_1 P^{\mathsf{T}}x = \begin{bmatrix}1_{n'} & 0_{m-n'}\end{bmatrix}G(\hat{\mu})\tilde{Q}Q^{T}y.$$

Together with equation (7.23) we get

$$\begin{bmatrix}1_{n'} & 0_{m-n'}\end{bmatrix}G(\hat{\mu})\left(R + \hat{\mu}\begin{bmatrix}Q_0^{\mathsf{T}}a_{k-q-2}\\ \sqrt{\gamma}e_0\end{bmatrix}e_{n'-1}^{\mathsf{T}}\right)P^{T}x = \begin{bmatrix}1_{n'} & 0_{m-n'}\end{bmatrix}G(\hat{\mu})\tilde{Q}Q^{T}y. \tag{7.33}$$

The solution of this linear system is cheap as the left hand side is upper triangular.

7.4 Refinement of standstill periods

In chapter 5 we described how to identify approximate locations of standstills from given data points. Now, as we also know how to enforce the corresponding boundary conditions we may bring both information together and suggest a method to identify exact standstill locations together with an approximation of their surrounding.

Let $S = [a, b]$ equal the approximated standstill interval. Furthermore, we assume for simplicity that S is in the middle of the movement. Now, we might define a simple knot sequence

$$\mathcal{T} := [\underbrace{a - l\delta, \ldots, a - l\delta}_{q+1 \text{ times}}, a - (l-1)\delta, \ldots, a - \delta, a, b, b + \delta, \ldots, b + (l-1)\delta, \underbrace{b + l\delta, \ldots, b + l\delta}_{q+1 \text{ times}}]$$

with given constants $\delta \in \mathbb{R}$ and $l \in \mathbb{N}$. Then the standstill is in the interval $[\tau_{q+l}, \tau_{q+l+1}]$.

Remark 7.34. It has to be taken care that the knot sequence lies completely within the time sequence of the movement, which requires $a - l\delta \geq t_0$ and $a + l\delta \leq t_{m-1}$. In addition we have to ensure that each knot interval contains at least one, better three or four, data points. Otherwise we risk the approximation problem becoming ill-conditioned (see Remark 4.35).

Using this knot sequence, we may approximate the data employing suitable boundary conditions and yield a spline approximation f. While doing so it is useful to weight the data points within the standstill such that

$$\sum_{i \in \{j \mid t_j \in S\}} w_i = \lambda$$

for given $\lambda \in \mathbb{R}$.

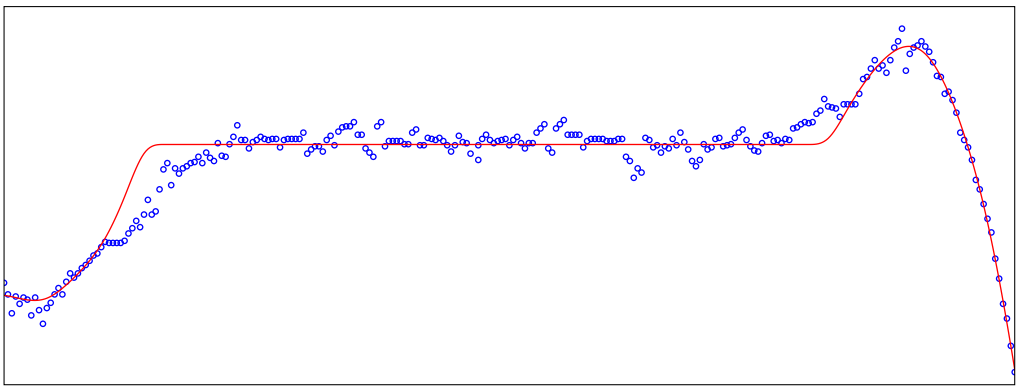


Figure 7.3: The standstill is slightly too long, thus the fitted curve diverges from the data points a few seconds before the actual standstill begin.

This way we ensure that the standstill always has the same effect, no matter how long it is. Failing this, long standstills would completely overweight the more relevant behaviour surrounding the standstill.

A first check on the quality of the standstill duration can be made by verifying

$$\|d_{l-1} - d_l\|_2^2 \geq \alpha \tag{7.35}$$

and

$$\left\|d_{q+l+1} - d_{q+l+2}\right\|_2^2 \geq \alpha \tag{7.36}$$

for a given constant α. According to equations (7.8) and (7.9) this norm of control point differences corresponds directly to the norm of f' on the intervals before and after the standstill. Thus, if the first condition above is violated, the movement before the standstill is so small that it is not significant. This indicates that we should decrease a and try again. Analogously, if the second condition is violated, we increase b and repeat the process.

If the data points within the standstill are trustworthy, it is likely that the standstill candidate S already was fairly exact and therefore the standstill should be well approximated. If, however, the standstill candidate was off, it has to be expected that the points within the standstill are of bad quality. Therefore, we will examine the region bordering the holding interval. Figure 7.3 shows a typical example: The first few data points bordering the beginning of the standstill are fitted well, while the points before those could clearly be better represented by the curve. To judge the quality of the standstill duration, it hence is best to examine a region close to, but not exactly bordering the standstill.

Let j_l such that $a = t_{j_l}$ and j_r such that $b = t_{j_r}$. We define the operators

$$Lf := \sum_{i=i_0}^{i_{\max}} \rho^* \left(f(t_{j_l - i}) - P_{j_l - i}\right)$$

and

$$Rf := \sum_{i=i_0}^{i_{\max}} \rho^* \left(f(t_{j_r + i}) - P_{j_r + i}\right)$$

for $i_0, i_{\max} \in \mathbb{N}$.

Now we will modify the standstill as long as we can reduce the values of Lf and Rf. To ensure that we do not diverge to much from the original standstill location, we will limit the absolute change to a certain amount. We find

Algorithm 7.2: Standstill refinement

Input: Standstill candidate $S = [t_{j_l}, t_{j_r}]$ with equations (7.35) and (7.36) enforced.
Maximal change θ.

Result: Refined standstill S and approximation f.

$i_l \leftarrow 0$
$i_r \leftarrow 0$
$f \leftarrow$ approximation with standstill $[t_{j_l}, t_{j_r}]$
for all $\sigma \in \{1, -1\}$ **do**
  repeat
    changed $\leftarrow$ false
    $f^* \leftarrow$ approximation with standstill $[t_{j_l+i_l-\sigma}, t_{j_r+i_r}]$
    if $Lf^* < Lf$ **and** $|i_l| \leq \theta$ **then**
      $f \leftarrow f^*$
      $i_l \leftarrow i_l - \sigma$
      changed $\leftarrow$ true
    end if
    $f^* \leftarrow$ approximation with standstill $[t_{j_l+i_l}, t_{j_r+i_r+\sigma}]$
    if $Rf^* < Rf$ **and** $|i_r| \leq \theta$ **then**
      $f \leftarrow f^*$
      $i_l \leftarrow i_r + \sigma$
      changed $\leftarrow$ true
    end if
  until changed = false
end for
$S \leftarrow [t_{j_l+i_l}, t_{j_r+i_r}]$

Note that this algorithm is a brute force local optimization method that does in no way guarantee that the best possible minimum is found. On the contrary, longer standstills usually mean better approximation results, which is why we first try to increase the standstill duration before trying to reduce it.

7.5 Speeding up the approximation

Due to the refinement steps stated above as well as the employment of robust smoothing methods, approximating one standstill requires solving a multitude of least squares problems. Therefore, it is worth to spend some time on speeding up the process of solving such a least squares problem. We already encountered two different numerical approaches to do so: Either by solving the normal equation

$$A^\mathsf{T} W A x = A^T W y$$

or by computing the QR-decomposition of A and using this decomposition to solve the problem directly.

If A is a $m \times n$ matrix with $m \geq n$, the normal projection approach requires us to compute the matrix product $A^\mathsf{T} A$. This can either be done with the conventional multiplication approach, taking $O(m^2 n)$ flops or using a fast matrix multiplication technique such as Strain's matrix multiplication, requiring $O(m^{2.807})$ flops [36, p. 31]. Besides that, the solution of the linear system needs about $O(n^3)$ computations. Similarly, computing the QR-decomposition also requires $O(m^2 n)$ operations.

Hence we may reduce the computational effort by a fair amount if we reduce m. In the presence of standstills this turns out to be surprisingly easy. All our standstill approximation problems lead to a minimization problem of the form

$$\min_x \sum_{i=0}^{m-1} w_i (y_i - f(t_i, x))^2$$

with one boundary condition being

$$f(x) = \text{const.} \qquad \text{for all } x \in [\tau_k, \tau_{k+1}].$$

Let $J = \{i \mid t_i \in [\tau_k, \tau_{k+1}]\}$. Splitting up the sum yields

$$\min_x \left(\sum_{i \notin J}^{m-1} w_i (y_i - f(t_i, x))^2 + \sum_{i \in J}^{m-1} w_i (y_i - f(\tau_k, x))^2 \right). \tag{7.37}$$

This can be simplified significantly, as the following lemma shows:

Lemma 7.38. *Given any function f, it follows that*

$$\sum_{i \in J} w_i (y_i - f(x))^2 = \left(\frac{\sum_{i \in J} w_i y_i}{\sum_{i \in J} w_i} - f(x) \right)^2 \sum_{i \in J} w_i + \kappa$$

for some $\kappa \in \mathbb{R}$.

Proof. With

$$\begin{aligned}\kappa &:= \sum_{i\in J} w_i(y_i - f(x))^2 - \left(\frac{\sum_{i\in J} w_i y_i}{\sum_{i\in J} w_i} - f(x)\right)^2 \sum_{i\in J} w_i \\ &= \sum_{i\in J} w_i(y_i^2 - 2f(x)y_i + f(x)^2) - \left(\left(\frac{\sum_{i\in J} w_i y_i}{\sum_{i\in J} w_i}\right)^2 - 2f(x)\frac{\sum_{i\in J} w_i y_i}{\sum_{i\in J} w_i} + f(x)^2\right)\sum_{i\in J} w_i \\ &= \sum_{i\in J} w_i y_i^2 - \frac{\left(\sum_{i\in J} w_i y_i\right)^2}{\sum_{i\in J} w_i}\end{aligned}$$

the result follows immediately. □

Thus the solution of equation (7.37) is identical to the solution of

$$\min_x \left(\sum_{i\notin J}^{m-1} w_i(y_i - f(t_i, x))^2 + \left(\frac{\sum_{i\in J} w_i y_i}{\sum_{i\in J} w_i} - f(\tau_k, x)\right)^2 \sum_{i\in J} w_i\right). \tag{7.39}$$

This amounts to replacing all points within the standstill with one point corresponding to their weighted average and giving this point the sum of all weights within the standstill as a new weight.

The resulting least squares problem has $m - |J|$ data points left and therefore might be solved in $O\left((m - |J|)^2 n\right)$ flops, whereas the computation of the weighted mean can be done in $O(|J|)$ steps. Especially for longer standstills it is a common situation that $m - |J| \ll |J|$, thus the reduction in flops is highly significant. Numerical experiments have shown that the execution time of the smoothing algorithm decreases by a factor of 4 for standstills of average length.

8 Filling the gaps

8.1 Local approximation

So far we have only considered how to approximate a spline that contains at most one standstill. The average aircraft movement, however, may often contain significantly more than one.

After having approximated the direct neighbourhood of the detected standstills it is now time to fill the gaps between those standstills with a suitable spline. As the general idea of robust smoothing splines has been explained in chapters 4 and 6 we just have to take care to choose suitable boundary conditions that allow a smooth transition of the standstill approximation.

Suppose we have a knot sequence

$$\mathcal{T} = [\ldots, \tau_{k-2}, \tau_{k-1}, \tau_k, \tau_{k+1}, \tau_{k+2}, \ldots]$$

with a standstill period in the interval $[\tau_k, \tau_{k+1}]$. Furthermore, suppose that the exact location c of the standstill is known. According to (7.4) we have

$$d_{k-q} = \cdots = d_k = c.$$

Due to the locality of splines one may easily see that the approximation $\hat{f}(t)$ only depends on c and $d_0, \ldots, d_{k-q-1}$ for $t \leq \tau_k$ and similarly only depends on c and $d_{k+1}, \ldots, d_{n-1}$ for $t \geq \tau_{k+1}$. Thus, if we assume c is known, the approximation for the whole spline splits up in two parts that are not depending on each other. In other words: The approximation problem can be solved for $t \leq \tau_k$ and $t \geq \tau_{k+1}$ separately.

This justifies the approach to first approximate the standstill locally and use this information to determine the standstill position. The full approximation problem then may be split into two by the standstill period. The only aspect we still have to discuss is how to translate the more restricting boundary conditions. We will take the limiting behaviour before and after the standstill from the local standstill approximation and translate it to the new, gap-filling approximation. Therefore, let

$$\mathcal{T}' = [\tau'_0, \ldots, \tau'_{k'} = \tau_k, \tau'_{k'+1} = \tau_{k+1}, \ldots, \tau'_{n'+q}]$$

be the knot sequence of the local approximation $l(x)$ and $d'_0, \ldots, d'_{n'}$ the corresponding control points. We will first discuss the behaviour before the standstill. To enforce either one of the known boundary conditions, we require

$$f'(\tau_k - \varepsilon) = l'(\tau_{k'} - \varepsilon)$$

for sufficiently small ε, which is

$$\lim_{\varepsilon \to 0} \frac{f'(\tau_k - \varepsilon)}{l'(\tau_{k'} - \varepsilon)} = 1. \tag{8.1}$$

According to equation (7.8) and using the recursive B-spline Definition 3.2 we have

$$f'(\tau_k - \varepsilon) = (d_k - d_{k-q-1})\frac{q}{\tau_k - \tau_{k-q}} N_{k-q,q-1}(\tau_k - \varepsilon).$$

This expression may easily be simplified:

Lemma 8.2. *For $q \geq 1$ and $\varepsilon < \tau_k - \tau_{k-1}$*

$$N_{k-q,q-1}(\tau_k - \varepsilon) = \frac{\varepsilon^{q-1}}{\prod_{j=1}^{q-1}(\tau_k - \tau_{k-j})}$$

and

$$N_{k+1,q-1}(\tau_{k+1} + \varepsilon) = \frac{\varepsilon^{q-1}}{\prod_{j=1}^{q-1}(\tau_{k+1+j} - \tau_{k+1})}.$$

Proof. For $q = 1$ the result follows directly from the definition of B-splines. Now assume that Lemma 8.2 is correct for $q - 1$. Then

$$\begin{aligned} N_{k-q,q-1}(\tau_k - \varepsilon) &= \frac{\tau_k - \varepsilon - \tau_{k-q}}{\tau_{k-1} - \tau_{k-q}} \underbrace{N_{k-q,q-2}(\tau_k - \varepsilon)}_{=0} + \frac{\varepsilon}{\tau_k - \tau_{k-q-1}} N_{k-q+1,q-2}(\tau_k - \varepsilon) \\ &= \frac{\varepsilon}{\tau_k - \tau_{k-q-1}} \frac{\varepsilon^{q-2}}{\prod_{j=1}^{q-2}(\tau_k - \tau_{k-j})} = \frac{\varepsilon^{q-1}}{\prod_{j=1}^{q-1}(\tau_k - \tau_{k-j})}. \end{aligned}$$

The second equation follows analogously. □

Thus

$$f'(\tau_k - \varepsilon) = (d_k - d_{k-q-1})q\frac{\varepsilon^q}{\prod_{j=1}^{q-1}(\tau_k - \tau_{k-j})}$$

and

$$l'(\tau_{k'} - \varepsilon) = (d'_{k'} - d'_{k'-q-1})q\frac{\varepsilon^q}{\prod_{j=1}^{q-1}(\tau'_{k'} - \tau'_{k'-j})}.$$

We yield

$$\lim_{\varepsilon \to 0} \frac{f'(\tau_k - \varepsilon)}{l'(\tau_{k'} - \varepsilon)} = \frac{d_k - d_{k-q-1}}{d'_{k'} - d'_{k'-q-1}} \prod_{j=1}^{q} \frac{\tau'_{k'} - \tau'_{k'-j}}{\tau_k - \tau_{k-j}}$$

and therefore

$$d_{k-q-1} = d_k - \left(d'_{k'} - d'_{k'-q-1}\right) \prod_{j=1}^{q} \frac{\tau_k - \tau_{k-j}}{\tau'_{k'} - \tau'_{k'-j}}. \tag{8.3}$$

The control point after the standstill may be computed similarly using equation (7.9):

$$\lim_{\varepsilon \to 0} \frac{f'(\tau_{k+1} + \varepsilon)}{l'(\tau_{k'+1} - \varepsilon)} = \frac{d_{k+1} - d_k}{d'_{k'+1} - d'_{k'}} \prod_{j=1}^{q} \frac{\tau'_{k'+1+j} - \tau'_{k'+1}}{\tau_{k+1+j} - \tau_{k+1}},$$

thus

$$d_{k+1} = (d'_{k'+1} - d'_{k'}) \prod_{j=1}^{q} \frac{\tau_{k+1+j} - \tau_{k+1}}{\tau'_{k'+1+j} - \tau'_{k'+1}} + d_k. \tag{8.4}$$

We just have shown how the boundary conditions of the local approximation translate to the control points of the spline segments between standstills. We gain

$$\mathbf{d}^*_\beta = \begin{bmatrix} c - \left(d'_{\beta,k'} - d'_{\beta,k'-q-1}\right) \prod_{j=1}^{q} \frac{\tau_k - \tau_{k-j}}{\tau'_{k'} - \tau'_{k'-j}} \\ c \\ \vdots \\ c \\ c + (d'_{\beta,k'+1} - d'_{\beta,k'}) \prod_{j=1}^{q} \frac{\tau_{k+1+j} - \tau_{k+1}}{\tau'_{k'+1+j} - \tau'_{k'+1}} \end{bmatrix}.$$

This splits the control point vector into

$$\mathbf{d}_\beta =: \left[\mathbf{d}^{(0)}_\beta, \mathbf{d}^*_\beta, \mathbf{d}^{(1)}_\beta\right]^\mathsf{T} = [[d_{\beta,0}, \ldots, d_{\beta,k-q-2}], [d_{\beta,k-q-1}, \ldots, d_{\beta,k+1}], [d_{\beta,k+2}, \ldots, d_{\beta,n-1}]]^\mathsf{T}$$

where the entries of $d^{(0)}_\beta$ are the control points before the standstill and the entries of $d^{(1)}_\beta$ the control points after the standstill. Similarly, we may split up the matrix $N_q(\mathbf{T}) := [N^{(0)}_\mathsf{T}, N^*_\mathsf{T}, N^{(1)}_\mathsf{T}]$ such that

$$N_q(\mathbf{T})\mathbf{d}_\beta := \left[N^{(0)}_\mathbf{T}, N^*_\mathbf{T}, N^{(1)}_\mathbf{T}\right] \begin{bmatrix} \mathbf{d}^{(0)}_\beta \\ \mathbf{d}^*_\beta \\ \mathbf{d}^{(1)}_\beta \end{bmatrix} = N^{(0)}_\mathbf{T}\mathbf{d}^{(0)}_\beta + N^*_\mathbf{T}\mathbf{d}^*_\beta + N^{(1)}_\mathbf{T}\mathbf{d}^{(1)}_\beta.$$

According to equations (6.4) and (6.5) the spline approximation problem without constraints in vector notation is

$$\min_{\mathbf{d}_\beta} \rho^*\left(\mathbf{P}_\beta - N_q(\mathbf{T})\mathbf{d}_\beta\right) + \mathbf{d}^\mathsf{T}_\beta G_2^\mathsf{T} B_q(\mathcal{T}, \lambda) G_2 \mathbf{d}_\beta.$$

This allows us to alter the optimization problem by incorporating the known control points:

$$\min_{\mathbf{d}^{(0)}_\beta, \mathbf{d}^{(1)}_\beta} \rho^*\left(\mathbf{P}_\beta - N^*_\mathbf{T}\mathbf{d}^*_\beta - N^{(0)}_\mathbf{T}\mathbf{d}^{(0)}_\beta - N^{(1)}_\mathbf{T}\mathbf{d}^{(1)}_\beta\right) + \begin{bmatrix} \mathbf{d}^{(0)}_\beta \\ \mathbf{d}^*_\beta \\ \mathbf{d}^{(1)}_\beta \end{bmatrix}^\mathsf{T} G_2^\mathsf{T} B_q(\mathcal{T}, \lambda) G_2 \begin{bmatrix} \mathbf{d}^{(0)}_\beta \\ \mathbf{d}^*_\beta \\ \mathbf{d}^{(1)}_\beta \end{bmatrix} \tag{8.5}$$

for $\beta \in \{x, y\}$.

By the locality of splines we note that $N^{(0)}_\mathsf{T}$ has non-zero rows only for the times that were recorded before the standstill. Likewise $N^{(1)}_\mathsf{T}$ has non-zero rows only for times recorded after the standstill. Specifically let

$$\mathbf{T}^{(0)} := \{t_i \mid t_i < \tau_k\}, \qquad \mathbf{T}^{(1)} := \{t_i \mid t_i > \tau_{k+1}\}$$

and let $\mathbf{P}^{(0)}_\beta, \mathbf{P}^{(1)}_\beta$ contain the corresponding data points. Then the term modelling the approximation accuracy from (8.5) becomes

$$\rho^*\left(\mathbf{P}^{(0)}_\beta - N^*_{\mathbf{T}^{(0)}}\mathbf{d}^*_\beta - N^{(0)}_{\mathbf{T}^{(0)}}\mathbf{d}^{(0)}_\beta\right) + \rho^*\left(\mathbf{P}^{(1)}_\beta - N^*_{\mathbf{T}^{(1)}}\mathbf{d}^*_\beta - N^{(1)}_{\mathbf{T}^{(1)}}\mathbf{d}^{(1)}_\beta\right) + c$$

for some $c \in \mathbb{R}$ and $\beta \in \{x, y\}$.

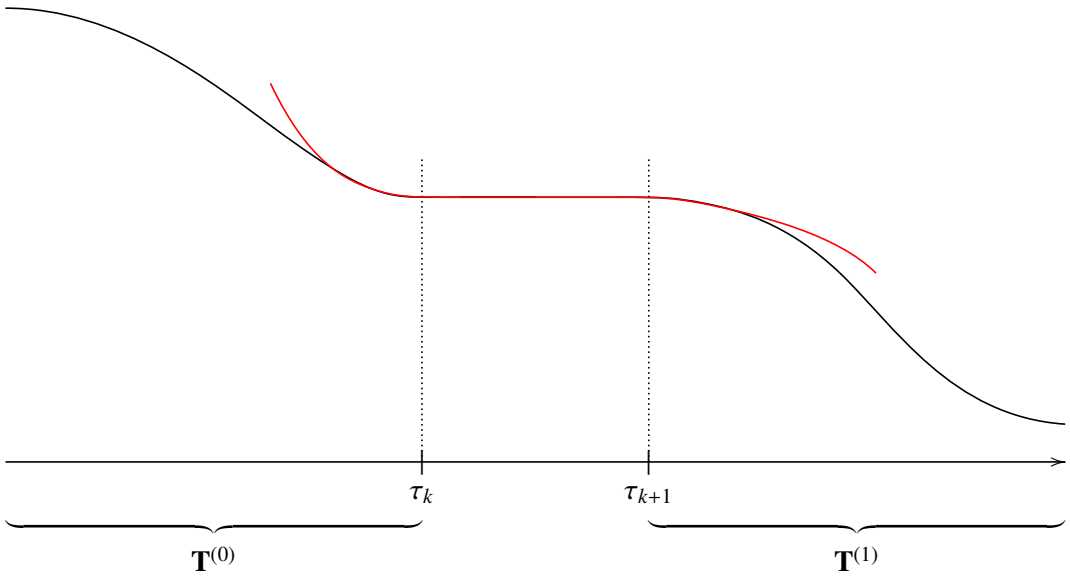


Figure 8.1: The local standstill approximation l (drawn in red) determines the behaviour of the curve close to the standstill $[\tau_k, \tau_{k+1}]$. The expressions relevant to the approximation left of the standstill are marked with the superscript (0), the expressions relevant to the approximation right of the standstill with (1).

As a next step we split up the smoothing matrix such that

$$G_2^\mathsf{T} B(\mathcal{T}, \lambda) G_2 =: \begin{bmatrix} A_{1,1} & A_{2,1}^\mathsf{T} & A_{3,1}^\mathsf{T} \\ A_{2,1} & A_{2,2} & A_{3,2}^\mathsf{T} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix}$$

and

$$A_{1,1} \in \mathbb{R}^{k-q-1 \times k-q-1}, \qquad A_{2,2} \in \mathbb{R}^{q+3,q+3}, \qquad A_{3,3} \in \mathbb{R}^{n-k-2}.$$

As $G_2^\mathsf{T} B(\mathcal{T}, \lambda) G_2$ is banded with a bandwidth of $q+2$ according to Remark 6.7 it follows that $A_{3,1} = 0$. Then

$$\begin{aligned} \begin{bmatrix} \mathbf{d}_\beta^{(0)} \\ \mathbf{d}_\beta^* \\ \mathbf{d}_\beta^{(1)} \end{bmatrix}^\mathsf{T} G_2^\mathsf{T} B_q(\mathcal{T}, \lambda) G_2 \begin{bmatrix} \mathbf{d}_\beta^{(0)} \\ \mathbf{d}_\beta^* \\ \mathbf{d}_\beta^{(1)} \end{bmatrix} &= \left[\mathbf{d}_\beta^{(0)\mathsf{T}}, \mathbf{d}_\beta^{*\mathsf{T}}, \mathbf{d}_\beta^{(1)\mathsf{T}} \right] \begin{bmatrix} A_{1,1} & A_{2,1}^\mathsf{T} & 0 \\ A_{2,1} & A_{2,2} & A_{3,2}^\mathsf{T} \\ 0 & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} \mathbf{d}_\beta^{(0)} \\ \mathbf{d}_\beta^* \\ \mathbf{d}_\beta^{(1)} \end{bmatrix} \\ &= d_\beta^{(0)\mathsf{T}} A_{1,1} d_\beta^{(0)} + 2 d_\beta^{*\mathsf{T}} A_{2,1} d_\beta^{(0)} + 2 d_\beta^{*\mathsf{T}} A_{3,2}^\mathsf{T} d_\beta^{(1)} + d_\beta^{(1)\mathsf{T}} A_{3,3} d_\beta^{(1)} + d_\beta^{*\mathsf{T}} A_{2,2} d_\beta^*. \end{aligned}$$

In summary the constrained optimization problem (8.5) may be split up into the two independent problems

$$\min_{d_\beta^{(0)}} \rho^* \left(\mathbf{P}_\beta^{(0)} - N_{\mathbf{T}^{(0)}}^* \mathbf{d}_\beta^* - N_{\mathbf{T}^{(0)}}^{(0)} \mathbf{d}_\beta^{(0)} \right) + \mathbf{d}_\beta^{(0)\mathsf{T}} A_{1,1} \mathbf{d}_\beta^{(0)} + 2 \mathbf{d}_\beta^{*\mathsf{T}} A_{2,1} \mathbf{d}_\beta^{(0)} \tag{8.6}$$

and

$$\min_{d_\beta^{(1)}} \rho^* \left(\mathbf{P}_\beta^{(1)} - N_{\mathbf{T}^{(1)}}^* \mathbf{d}_\beta^* - N_{\mathbf{T}^{(1)}}^{(1)} \mathbf{d}_\beta^{(1)} \right) + \mathbf{d}_\beta^{(1)\mathsf{T}} A_{3,3} \mathbf{d}_\beta^{(1)} + 2 \mathbf{d}_\beta^{*\mathsf{T}} A_{3,2}^\mathsf{T} \mathbf{d}_\beta^{(1)} \tag{8.7}$$

for $\beta \in \{x, y\}$.

Note that both (8.6) and (8.7) are convex problems: ρ^* of a linear function is convex and the new smoothing penalty is convex as a shifted partial function of the original convex smoothing penalty.

When applying the IRLS algorithm, which is by now well known, we gain a series of weighted least squares problems. Those might be solved by referring to equation (4.6). The required gradients of the smoothing penalties are

$$2A_{1,1}\mathbf{d}_\beta^{(0)} + 2A_{1,1}^\mathsf{T}\mathbf{d}_\beta^*$$

and

$$2A_{3,3}\mathbf{d}_\beta^{(1)} + 2A_{3,2}\mathbf{d}_\beta^*.$$

Thus, we obtain the linear systems

$$\left(N_{\mathbf{T}^{(0)}}^{(0)\mathsf{T}} W N_{\mathbf{T}^{(0)}}^{(0)} + A_{1,1}\right)\mathbf{d}_\beta^{(0)} = N_{\mathbf{T}^{(0)}}^{(0)\mathsf{T}} W\left(\mathbf{P}_\beta^{(0)} - N_{\mathbf{T}^{(0)}}^* \mathbf{d}_\beta^*\right) - A_{2,1}^\mathsf{T}\mathbf{d}_\beta^* \tag{8.8}$$

and

$$\left(N_{\mathbf{T}^{(1)}}^{(1)\mathsf{T}} W N_{\mathbf{T}^{(1)}}^{(1)} + A_{3,3}\right)\mathbf{d}_\beta^{(1)} + = N_{\mathbf{T}^{(1)}}^{(1)\mathsf{T}} W\left(\mathbf{P}_\beta^{(1)} - N_{\mathbf{T}^{(1)}}^* \mathbf{d}_\beta^*\right) - A_{3,2}\mathbf{d}_\beta^*. \tag{8.9}$$

We just derived the concept to incorporate the correct boundary conditions into the approximation problem in the presence of one standstill. In real applications it is likely that we will have more than one standstill. However, the same concept could be applied if the movement part is surrounded by two standstill periods. This is rather a matter of strict notation than of mathematical insight and thus will be omitted here.

8.2 Choosing an optimal knot sequence

Good knot placement is of central importance for the quality of the approximation. If there are too few or badly placed knots, the smoothing spline will not be able to reproduce all features of the given data set. If, on the other hand, too many knots are used, the approximated spline will be less efficient to store and handle in all further computations. Thus, we have to find a good trade-off between the amount of knots and the quality of the approximation.

It is a very hard problem to find an optimal number of knots and knot sequence. Consequently, the best we may hope for is to find an approximation to an optimal knot sequence. A thorough examination of the topic has been conducted by de Boor in his *Practical Guide to Splines* [10, pp. 156 – 161]. We will repeat his ideas and make some small amendments to accommodate the concepts used in this thesis. The most notable of those is the generalization to curves in $\mathbb{R}^l$ instead of $\mathbb{R}$ as well as a small modification that allows to alter the required accuracy over the time domain.

First, we remember that we denoted the linear space spanned by the B-spline basis function $N_{0,q}(\cdot \mid \mathcal{T}), \ldots, N_{n-1,q}(\cdot \mid \mathcal{T})$ by $\mathbb{S}_q(\mathcal{T})$. For a spline with values in $\mathbb{R}^l$ all that we do is to allow the coefficients of the linear combination to lie in this l-dimensional space. It is convenient to introduce a separate notation:

$$\mathbb{S}_q^l(\mathcal{T}) := \left\{ \sum_{i=0}^{n-1} d_i N_{i,q}(t \mid \mathcal{T}) \;\middle|\; d_i \in \mathbb{R}^l \right\}. \tag{8.10}$$

For now we will assume that the function $g\colon \mathcal{T} \to \mathbb{R}^l$ to be approximated is known. The selection of the knot sequence aims at giving the approximation process all the freedom that is required but not to offer too many degrees of freedom. In other words, it is essential that the space $\mathbb{S}_q^l(\mathcal{T})$ contains a function that is close to g. This motivates

Definition 8.11.

$$\operatorname{dist}_I\left(g, \mathbb{S}_q^l(\mathcal{T})\right) := \min_{f \in \mathbb{S}_q(\mathcal{T})} \max_{x \in I} \|f(x) - g(x)\|_2$$

for some interval $I \subset \mathcal{T}$.

The most fundamental requirement is that the knot sequence $\mathcal{T}$ to be chosen has a minimal number of knots while satisfying

$$\operatorname{dist}_{\mathcal{T}}\left(g, \mathbb{S}_q^l(\mathcal{T}))\right) \leq C \tag{8.12}$$

for some value $C \in \mathbb{R}$. However, in our case it is of advantage to be able to let the required accuracy vary over time. Therefore, we define a function $\alpha\colon \mathcal{T} \to \mathbb{R}_+$ and restate (8.12) as

$$\min_{f \in \mathbb{S}_q(\mathcal{T})} \max_{x \in \mathcal{T}} \alpha(x)\|f(x) - g(x)\|_2 \leq 1. \tag{8.13}$$

Note that with $\alpha \equiv \frac{1}{C}$ this is equivalent to (8.12).

This concept may be used to model different accuracy requirements in different movement phases: As the movements in flight happen on a significantly larger scale, we may reduce the accuracy in these phases.

Finding a knot sequence $\mathcal{T}$ that fulfils (8.13) with an optimal number of knots is rather challenging. We can not hope to give such an exact solution without an excessive amount of computational efforts. Therefore, we will make some simplifications that will result in a knot sequence that fulfils those conditions at least approximately.

The first of those amendments is to look at an upper bound of the distance function of Definition 8.11. For this we will make use of the following result shown by de Boor in [12] and [10, p. 156]:

Lemma 8.14 (de Boor, 1973 [12])**.** *There exists a constant c_q such that for any given $q+1$-times differentiable function $g\colon [a,b] \to \mathbb{R}$ and a knot sequence $\mathcal{T} = [\tau_0, \ldots, \tau_{n+q}]$ with $\tau_0 = a$ and $\tau_{n+q} = b$ there exists $f \in \mathbb{S}_q(\mathcal{T})$ such that*

$$\max_{x \in [\tau_i, \tau_{i+1}]} |f(x) - g(x)| \leq c_q (\tau_{i+q} - \tau_{i-q+1})^{q+1} \max_{x \in [\tau_{i-q+1}, \tau_{i+q}]} \left|g^{(q+1)}(x)\right| \tag{8.15}$$

holds for all $i = q, \ldots, n-1$.

Note that due to the $q+1$-fold boundary knots of the knot sequence $\mathcal{T}$ it is not necessary that the bound holds for the remaining knot intervals.

As an addition to the work of de Boor we may generalize this result neatly to the l-dimensional case:

Corollary 8.16. *There exists a constant c_q such that for any given $q+1$-times differentiable function $g\colon [a,b] \to \mathbb{R}^l$ and a knot sequence $\mathcal{T} = [\tau_0, \dots, \tau_{n+q}]$, $\tau_0 = a$ and $\tau_{n+q} = b$ there exists a function $f \in \mathbb{S}_q^l(\mathcal{T})$ such that*

$$\max_{x\in[\tau_i,\tau_{i+1}]} \|f(x) - g(x)\|_2 \le \sqrt{l}\, c_q |J_i|^{q+1} \max_{x\in J_i} \left\| g^{(q+1)}(x) \right\|_2$$

with

$$J_i := [\tau_{i-q+1}, \tau_{i+q}], \qquad |J_i| := \tau_{i+q} - \tau_{i-q+1}$$

holds for all $i = q, \dots, n-1$.

Proof. Let $g = [g_0, \dots, g_{l-1}]^\mathsf{T}$ and $f_i \in \mathbb{S}_q(\mathcal{T})$ a function satisfying (8.15) for g_i. We define $f := [f_0, \dots, f_{l-1}]^\mathsf{T} \in \mathbb{S}_q^l(\mathcal{T})$. For $i = q, \dots, n-1$ we find that

$$\begin{aligned}\sum_{j=0}^{l-1} \max_{x\in J_i} \left(g_j^{(q+1)}(x)\right)^2 &\le l \max_{j=\{0,\dots,i-1\}} \max_{x\in J_i} \left(g_j^{(q+1)}(x)\right)^2 \\ &\le l \max_{x\in J_i} \sum_{j=0}^{l-1} \left(g_j^{(q+1)}(x)\right)^2 = l \max_{x\in J_i} \left\| g^{(q+1)}(x) \right\|_2^2.\end{aligned}$$

Hence

$$\begin{aligned}\max_{x\in[\tau_i,\tau_{i+1}]} \|f(x) - g(x)\|_2^2 &= \max_{x\in[\tau_i,\tau_{i+1}]} \sum_{j=0}^{l-1} \left((f_j(x) - g_j(x)\right)^2 \\ &\le \sum_{j=0}^{l-1} \max_{x\in[\tau_i,\tau_{i+1}]} \left(f_j(x) - g_j(x)\right)^2 \\ &\le c_q^2 |J_i|^{2(q+1)} \sum_{j=0}^{l-1} \max_{x\in J_i} \left(g_j^{(q+1)}(x)\right)^2 \\ &\le l c_q^2 |J_i|^{2(q+1)} \max_{x\in J_i} \left\| g_j^{(q+1)}(x) \right\|_2^2\end{aligned}$$

by applying Lemma 8.14. The result follows by taking the root of the derived bound. □

We may use Corollary 8.16 to give a slightly stricter condition on our knot sequence that obviously ensures that equation (8.12) is fulfilled:

$$\max_{i=q,\dots,n-1} \left(|J_i|^{q+1} \max_{x\in J_i} \left\| g^{(q+1)}(x) \right\|_2 \right) \le \frac{C}{c_q \sqrt{l}}.$$

Letting the accuracy vary over time as in (8.13) does not allow to use the bound above without further simplifications. However, employing another coarse estimate we find

$$\begin{aligned}&\max_{x\in[\tau_i,\tau_{i+1}]} \alpha(x) \|g(x) - f(x)\|_2 \\ &\quad\le \max_{x\in[\tau_i,\tau_{i+1}]} \alpha(x) \cdot \max_{x\in[\tau_i,\tau_{i+1}]} \|g(x) - f(x)\|_2 \\ &\quad\le \sqrt{l}\, c_q |J_i|^{q+1} \max_{x\in[\tau_i,\tau_{i+1}]} \alpha(x) \cdot \max_{x\in J_i} \left\| g^{(q+1)}(x) \right\|_2\end{aligned}$$

for $i = q, \dots, n-1$ and hence we may replace (8.13) by

$$\max_{i=q,\dots,n-1} \left(|J_i|^{q+1} \max_{x \in [\tau_i, \tau_{i+1}]} \alpha(x) \cdot \max_{x \in J_i} \left\| g^{(q+1)}(x) \right\|_2 \right) \leq \frac{1}{c_q \sqrt{l}}, \tag{8.17}$$

or, taking the $q+1$-th root, by

$$\max_{i=q,\dots,n-1} \left(|J_i| \max_{x \in [\tau_i, \tau_{i+1}]} \sqrt[q+1]{\alpha(x)} \cdot \max_{x \in J_i} \left\| g^{(q+1)}(x) \right\|_2^{\frac{1}{q+1}} \right) \leq \left(c_q^2 l \right)^{-\frac{1}{2(q+1)}}. \tag{8.18}$$

Up to now we have replaced the boundary condition of the approximation problem by the more restrictive condition (8.18). To gain a reasonable fast algorithm we will have to sacrifice strict mathematical accuracy with an approximated approach. De Boor claims that "we can not hope to place each knot optimally. We can only hope to obtain an optimal knot distribution or density." [10, p. 157] One way to do so is to note that for $\max_i(\tau_{i+1} - \tau_i) \to 0$ asymptotically

$$|J_i| \max_{x \in [\tau_i, \tau_{i+1}]} \sqrt[q+1]{\alpha(x)} \cdot \max_{x \in J_i} \left\| g^{(q+1)}(x) \right\|_2^{\frac{1}{q+1}} \approx \int_{J_i} \left(\alpha(x) \left\| g^{(q+1)}(x) \right\|_2 \right)^{\frac{1}{q+1}} \mathbf{d}x.$$

It follows that an asymptotically equivalent way to bound equation (8.18) is to bound

$$\max_{i=q,\dots,n-1} \int_{\tau_{i-q+1}}^{\tau_{i+q}} \left(\alpha(x) \left\| g^{(q+1)}(x) \right\|_2 \right)^{\frac{1}{q+1}} \mathbf{d}x \leq C \tag{8.19}$$

for some constant C. In practice one will choose C by trial and error. Thus the actual meaning of the constant is nice to know but not essential for the method itself. De Boor now assumes that all knots have q-fold multiplicity, which allows further simplifications. We will show that the same result can be achieved without using multiple knots:

Let

$$H(a,b) := \int_a^b \left(\alpha(x) \left\| g^{(q+1)}(x) \right\|_2 \right)^{\frac{1}{q+1}} \mathbf{d}x.$$

Theorem 8.20. *For given $n \in \mathbb{N}, \tau_0, \tau_{n+q} \in \mathbb{R}$*

$$\min_{\mathcal{T}} \max_{i=q,\dots,n-1} H(\tau_{i-q+1}, \tau_{i+q}) = \frac{2q-1}{n-q} H(\tau_0, \tau_{n+q})$$

where $\mathcal{T}$ iterates over all knot sequences with $q+1$-fold boundary knots at τ_0 and τ_{n+q}. This minimum is taken for $\mathcal{T}$ such that

$$H(\tau_i, \tau_{i+1}) = \frac{1}{n-q} H(\tau_0, \tau_{n+q})$$

for $i = q, \dots, n-1$.

Proof. Due to the additivity of integration it follows that

$$H(\tau_{i-q+1}, \tau_{i+q}) = \sum_{j=i-q+1}^{i+q-1} H(\tau_j, \tau_{j+1}).$$

Then

$$\begin{aligned}\max_{i=q,\dots,n-1} H(\tau_{i-q+1},\tau_{i+q}) &\geq \min_{i=2q-1,\dots,n-q} \sum_{j=i-q+1}^{i+q-1} H(\tau_j,\tau_{j+1}) \\ &\geq (2q-1) \min_{i=2q-1,\dots,n-q} \min_{j=i-q+1,\dots,i+q-1} H(\tau_j,\tau_{j+1}) \\ &= (2q-1) \min_{i=q,\dots,n-1} H(\tau_i,\tau_{i+1}). \end{aligned} \tag{8.21}$$

As the integrand of $H(a,b)$ is monotonically decreasing in a and monotonically increasing in b we may find knots $\tau_{q+1},\dots,\tau_{n-1}$. So there is a constant c with $c = H(\tau_i,\tau_{i+1})$ for all $i = q,\dots,n-1$. Then

$$H(\tau_0,\tau_{n+q}) = \sum_{i=q}^{n-1} H(\tau_i,\tau_{i+1}) = (n-q)c$$

and thus

$$c = \frac{H(\tau_0,\tau_{n+q})}{n-q}.$$

Furthermore

$$\max_{i=q,\dots,n-1} H(\tau_{i-q+1},\tau_{i+q}) = \max_{i=q,\dots,n-1} \sum_{j=i-q+1}^{i+q-1} H(\tau_j,\tau_{j+1}) = (2q-1)c.$$

and

$$(2q-1) \min_{i=q,\dots,n-1} H(\tau_i,\tau_{i+1}) = (2q-1)c.$$

Together with equation (8.21) we have shown that the given knot sequence minimizes (8.19). □

Theorem 8.20 gives us a recipe for our approximately optimal knot sequence if we can invert

$$H_{\tau_0}(x) := H(\tau_0,x) = \int_{\tau_0}^{x} \left(\alpha(t)\left\|g^{(q+1)}(t)\right\|_2\right)^{\frac{1}{q+1}} \mathbf{d}t.$$

Corollary 8.22. *For given $n \in \mathbb{N}, \tau_0, \tau_{n+q} \in \mathbb{R}$ an optimal knot sequence in the sense that* (8.19) *is minimized is given by $\tau_1 = \cdots = \tau_q := \tau_0$, $\tau_n = \cdots = \tau_{n+q-1} := \tau_{n+q}$ and*

$$\tau_i := H_{\tau_0}^{-1}\left(\frac{i-q}{n-q} H_{\tau_0}(\tau_{n+q})\right)$$

for $i = q+1,\dots,n-1$.

Note that de Boor's discussion on the topic concludes with the same result under the condition that $l = 1$ and $\alpha(t) \equiv const$.

From here all that is left to do is to find a method to invert $H_{\tau_0}(x)$. The method presented for this purpose will be identical to the algorithm proposed by de Boor [10, pp. 158 – 159] since the small generalizations introduced will not be important anymore.

Inverting H_{τ_0} will be easy, if $\alpha(x)\left\|g^{(q+1)}(x)\right\|_2$ is a piecewise constant function. This is the case, if both α and $g^{(q+1)}$ are piecewise constant. As α is just a tuning parameter, we may replace it by a piecewise constant approximation without too much hassle.

In our approximation context the function g is not known a priori but is a result of the smoothing process. Thus, before determining the optimal knot sequence we have to find an approximation of $g^{(q+1)}$. One way to do this would be to approximate the data with a function g of order $q+2$ and then use the $q+1$-th derivative of this function. De Boor suggested another way that does not require to raise the order of the approximation [11].

He recommends approximating g with a smoothing spline f of degree q with a suitable knot sequence $\mathcal{T}$ (i.e. by choosing equidistant knots). Then $g^{(q)}$ is a piecewise constant function that can easily be determined from the control points of the first approximation using (3.7). Specifically we obtain d_i' such that

$$g^{(q)}(x) \approx \sum_{i=0}^{n-q-1} d_i' N_{i,0}(x \mid \mathcal{T}^{(q)}). \tag{8.23}$$

De Boor now approximates this function by a spline of degree two and uses $g^{(q+1)}$ as the derivative of this spline. The approximation scheme he proposes is to fit a parabola that interpolates (8.23) at the sites $\frac{1}{2}(\tau_{i-1}^{(q)} + \tau_i^{(q)})$, $\frac{1}{2}(\tau_i^{(q)} + \tau_{i+1}^{(q)})$ and $\frac{1}{2}(\tau_{i+1}^{(q)} + \tau_{i+2}^{(q)})$ for each knot interval i. The slope of this parabola at $\frac{1}{8}(\tau_{i-1}^{(q)} + 3\tau_i^{(q)} + 3\tau_{i+1}^{(q)} + \tau_{i+2}^{(q)})$ is then chosen as value for $g^{(q+1)}$ in the knot interval $[\tau_i, \tau_{i+1})$. Including boundary effects this approach yields [12]

$$g^{(q+1)}(x) \approx \hat{g}^{(q+1)}(x) := \begin{cases} 2\frac{d_1' - d_0'}{\tau_2^{(q)} - \tau_0^{(q)}}, & \text{if } x \in \left[\tau_0^{(q)}, \tau_1^{(q)}\right), \\ \frac{d_i' - d_{i-1}'}{\tau_{i+1}^{(q)} - \tau_{i-1}^{(q)}} + \frac{d_{i+1}' - d_i'}{\tau_{i+2}^{(q)} - \tau_i^{(q)}}, & \text{if } x \in \left[\tau_i^{(q)}, \tau_{i+1}^{(q)}\right),\ i \in \{1, \dots, n-2\}, \\ 2\frac{d_n - d_{n-1}}{\tau_n^{(q)} - \tau_{n-2}^{(q)}}, & \text{if } x \in \left[\tau_{n-1}^{(q)}, \tau_n^{(q)}\right]. \end{cases}$$

This way we gain a piecewise constant approximation for the function $g^{(q+1)}$ and thus trivially also for $\left\|g^{(q+1)}\right\|$.

We will require

$$\alpha := \sum_{i=0}^{n-q-1} \alpha_i N_{i,0}(x \mid \mathcal{T})$$

to have the same knot vector as the initial spline approximation f. This requirement is no restriction: We could always insert knots into the break sequence of $\hat{g}^{(q+1)}$ as well as α until both sequences are equal. Furthermore we note that $\mathcal{T}^{(q)}$ equals the knot vector $\mathcal{T}$ with the first and last q knots removed. With

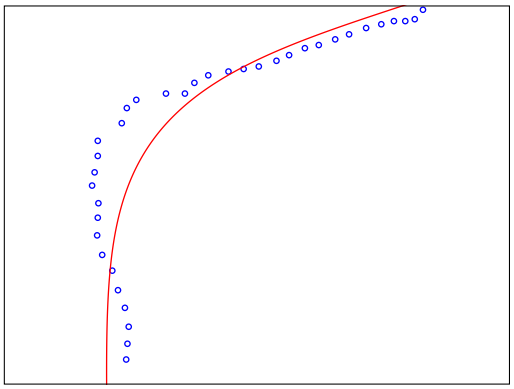
$$h_i := \sum_{j=q}^{i} (\tau_{j+1} - \tau_j) \left(\alpha_j \left\| \hat{g}^{(q+1)} \left(\frac{1}{2}(\tau_j + \tau_{j+1}) \right) \right\| \right)^{\frac{1}{q+1}}$$

we gain

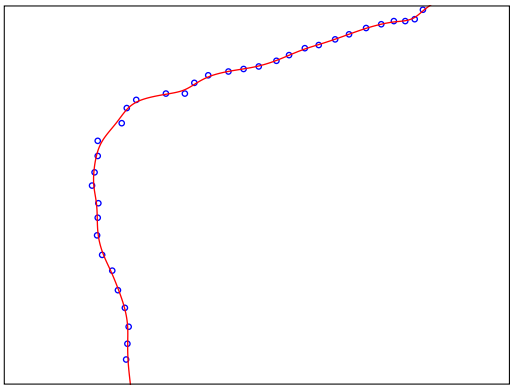
$$H_{\tau_0}(x) \approx \sum_{i=0}^{n-1} h_i N_{i,1}(x \mid \mathcal{T}). \tag{8.24}$$

A spline curve of degree one may be inverted by exchanging knots and control points while taking care of multiple boundary knots. Let

$$\mathcal{S} := [h_q, h_q, \dots, h_n, h_n].$$



(a) The fitted spline clearly diverges from the data points - the smoothing parameter is too large.



(b) The fitted spline oscillates and picks up behaviour clearly produced by noise.

Figure 8.2: Examples of the same part of a curve fitted with different smoothing parameters.

Then

$$H_{\tau_0}^{-1}(x) \approx \sum_{i=0}^{n-1} \tau_{i+q} N_{i,1}(x \mid \mathcal{S}). \tag{8.25}$$

An approximated optimal knot sequence then may be found using Corollary 8.22.

If required, we may easily apply this method to the partial approximation problems discussed in chapter 8.1. Only slight modifications of the h_i are necessary to deal with the fact that there are no multiple boundary knots.

8.3 A side note to the selection of the smoothing parameter

The smoothness of smoothing splines may be adjusted by the smoothing parameter as described in chapter 6. One important question for our approximation process is how to choose a smoothing parameter in a way that it does not force the fitted spline to disregard certain features in the curve. On the other hand it has to be ensured that the random error of the approximation does not influence the fitted approximation unduly. Examples of both situations are shown in Figure 8.2.

The selection of suitable smoothing parameters has been a topic of intensive research. Quite a few methods rely on Generalized Cross Validation [19] or Generalized Maximum Likelihood [69] to estimate a data-driven global smoothing parameter. A thorough introduction may be found in *Spline Models for Observational Data* by Grace Wahba [70].

In our situation a global smoothing parameter certainly is not able to model the varying accuracy present. Data driven approaches to select the smoothing parameter also have been investigated, for example by Pintore, Speckman and Holmes [59], Storlie, Bondell and Reich [64] as well as by Liu and Guo [51]. Especially Pintore et al. [59] also impose the restriction of piecewise constant λ. This greatly improves the computational efforts required to solve the actual smoothing spline problem. One drawback of those approaches is that they usually focus on dealing with varying roughness of the function to be approximated. For us, they should rather treat varying quality in the data observed.

It could be considered modifying one of their approaches or creating a new version that is suitable for our kind of problem. We would have to gain a data driven algorithm to select the smoothing parameter. This might raise the approximation quality further. However, the data offers much information in the first place that can be used to choose the smoothing parameter: The location of standstills, the approximated velocity, the phase of aircraft movement, the placement of the data points. Using fixed parameters employing only those information already yields good results, so we will spend no additional effort on researching this problem more thoroughly.

9 Numerical results and conclusion

Throughout this thesis we developed an approximation framework that allows the approximation of vehicle and aircraft movements under boundary conditions that enforce physically correct movement behaviour. Special care was taken to detect and approximate standstill periods suitably.

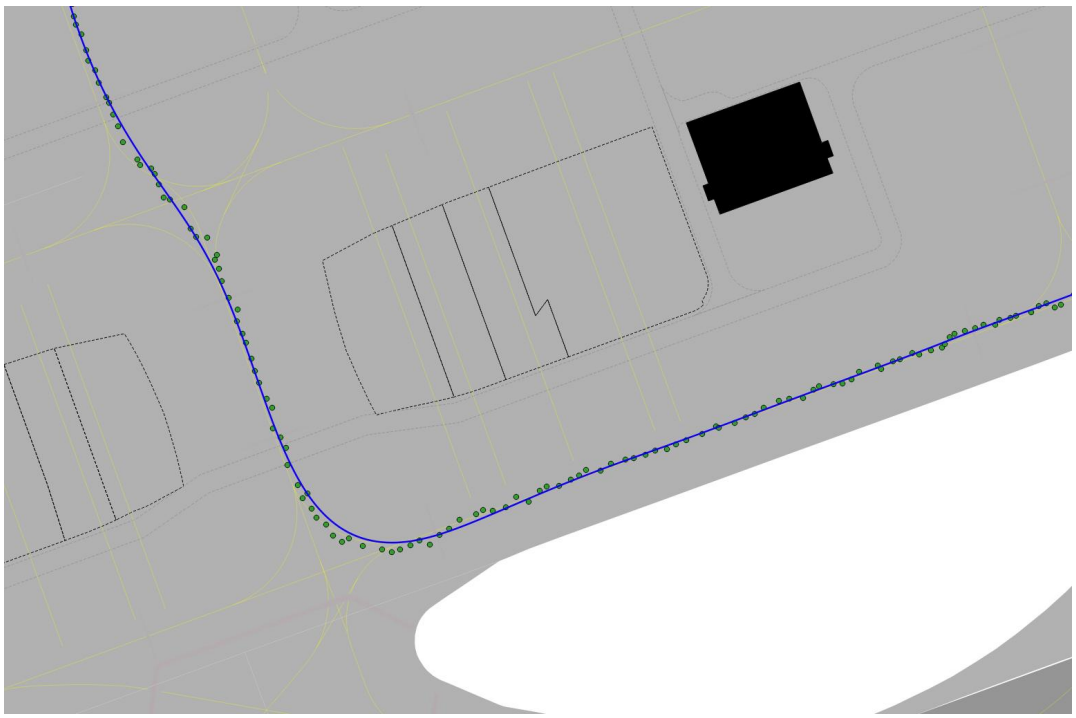
All algorithms presented have been implemented and tested as prototypes. The transfer of those prototypical implementations into Fraport's productive system currently is under way. Hence it is no surprise that large scale demonstration of the achieved results will only be possible after implementation has been completed. In the following we will highlight some examples that show the good performance of the developed algorithms. We will compare the newly achieved results with the smoothing algorithms used before this thesis. Those practically amount to using standard least squares with variable spaced knots and without further boundary conditions.

Figure 9.1 depicts the usual, smooth movement. It can be plainly seen that the new approximation yields slightly better approximation results. As all data points are of good quality, the reason for this is to be sought elsewhere: As the robust regression is able to handle outliers better, the smoothing parameter could be lowered without risking the approximation to be overly affected by outliers. As a result the fit adapts better throughout the curves.

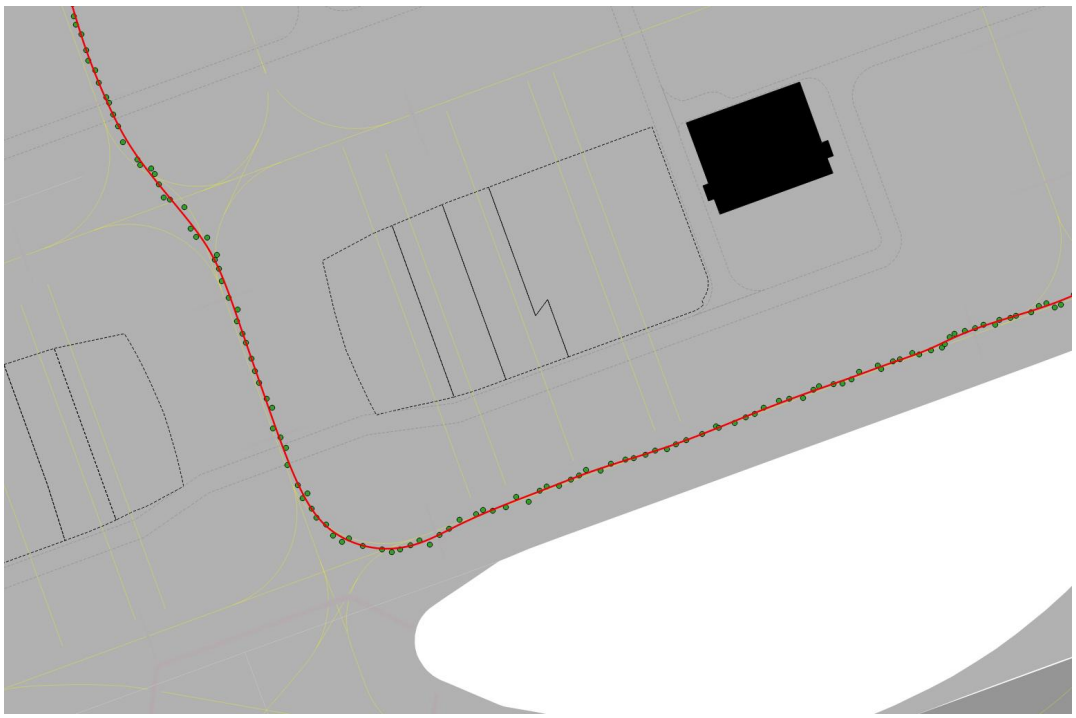
The effect achieved by employing robust smoothing methods can be seen in Figure 9.2. For better comparison the new approximation has been conducted without adding standstill boundary conditions. As the smoothing parameter previously had to be utilized to reduce oscillation caused by outliers in the data, correct details in the data could not be picked up by the approximation. The newly developed algorithms now handle outliers using robust smoothing methods. One should note that the maximum error during the standstill is smaller while on the same time the actual movement is fitted better.

Figures 9.3 and 9.4 demonstrate the use of the derived standstill boundary conditions. The approximation in Figure 9.3 mainly benefits from the smooth trajectory and exact location of the actual standstill. When looking at Figure 9.4 the significant increase in approximation quality becomes obvious. Here the enforced standstill causes the approximation to be at the centre of the aircrafts parking position. The outliers previously present do not pose a significant problem anymore. Secondly the approximation of the pushback procedure resembles the correct movement of the aircraft. Apart from a good approximation this also allows to detect the position and time where the pushback tractor is decoupled from the aircraft. From an operational perspective this is an important timestamp in aircraft processing that now can be determined with high accuracy.

We spent some effort to split up the approximation problem into single parts between standstills and also found a way to speed up the approximation of standstill significantly. This work clearly pays off in the approximation speed: On average it took about 0.4 seconds to approximate one aircraft movement without applying boundary conditions. If boundary conditions are added and

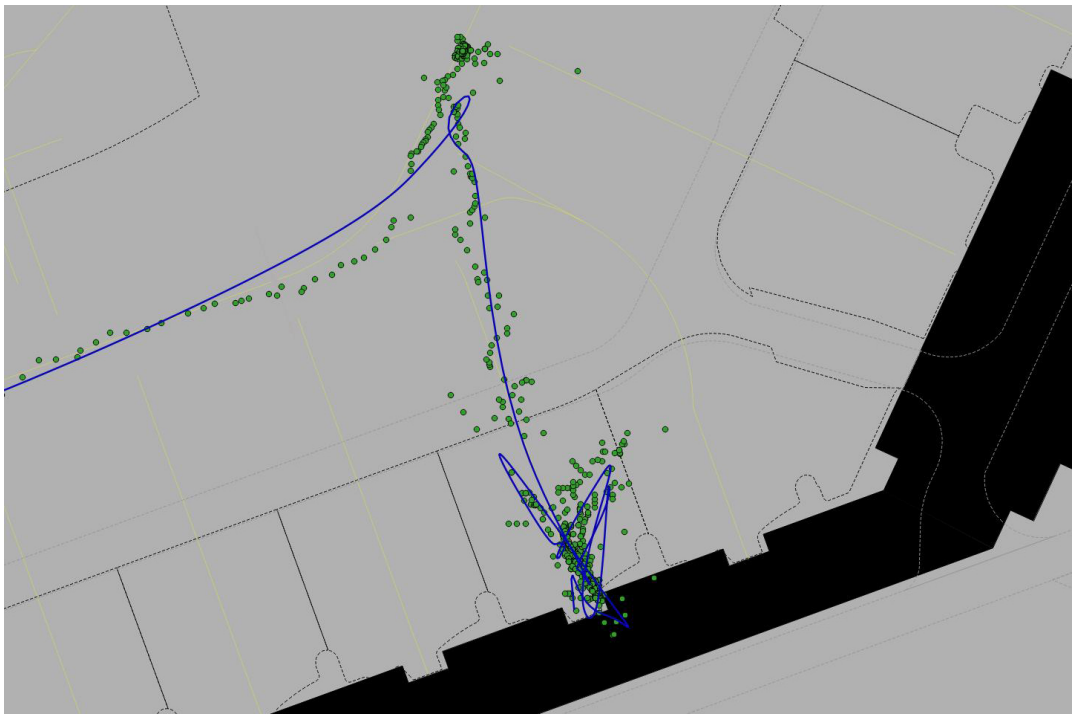


(a) Fraport's original least-squares approximation.

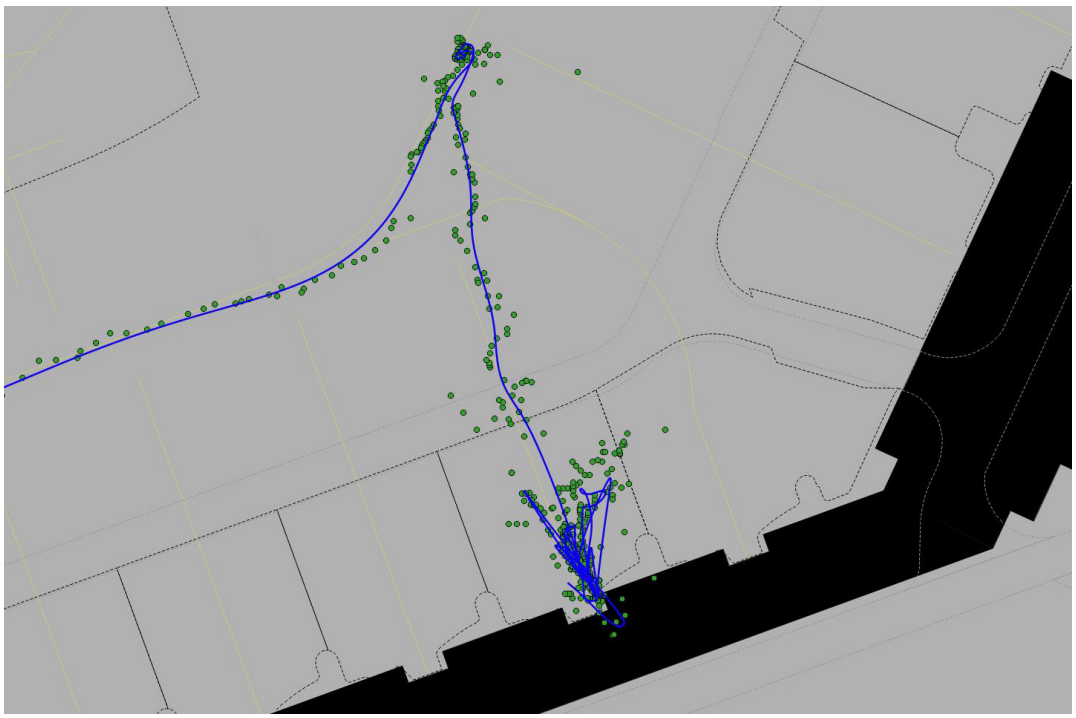


(b) Robust smoothing algorithm presented in this thesis.

Figure 9.1: Due to the robust fitting algorithm, the smoothing parameter could be lowered and thus a better adaption to the original data points could be achieved.

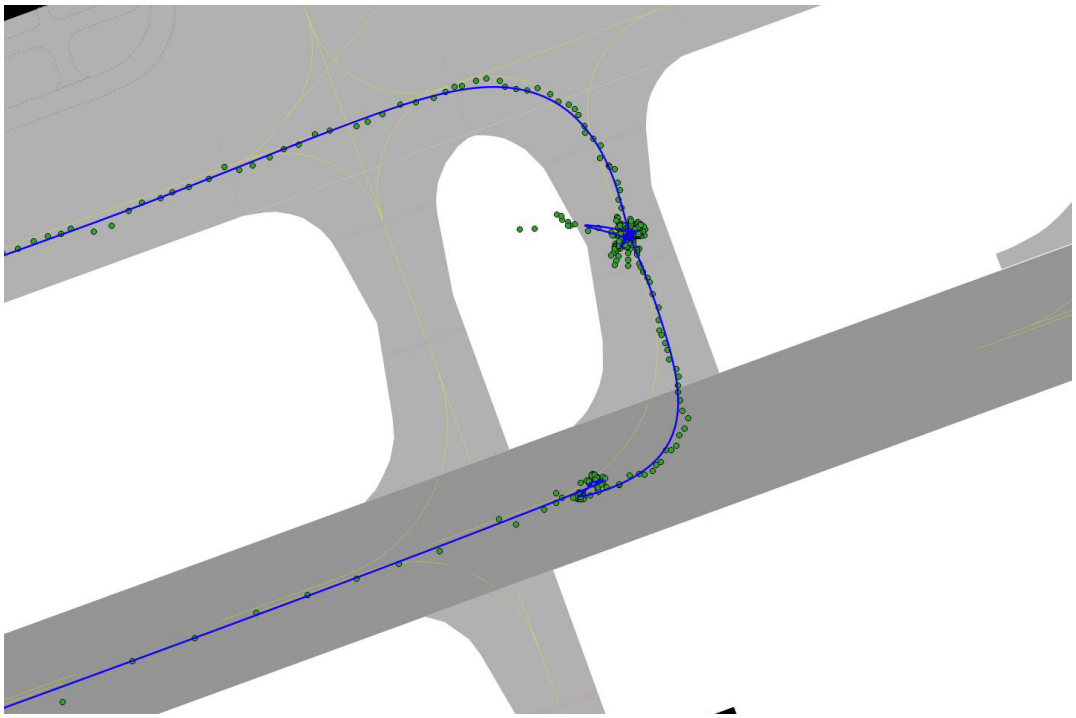


(a) Fraport's original least-squares approximation.

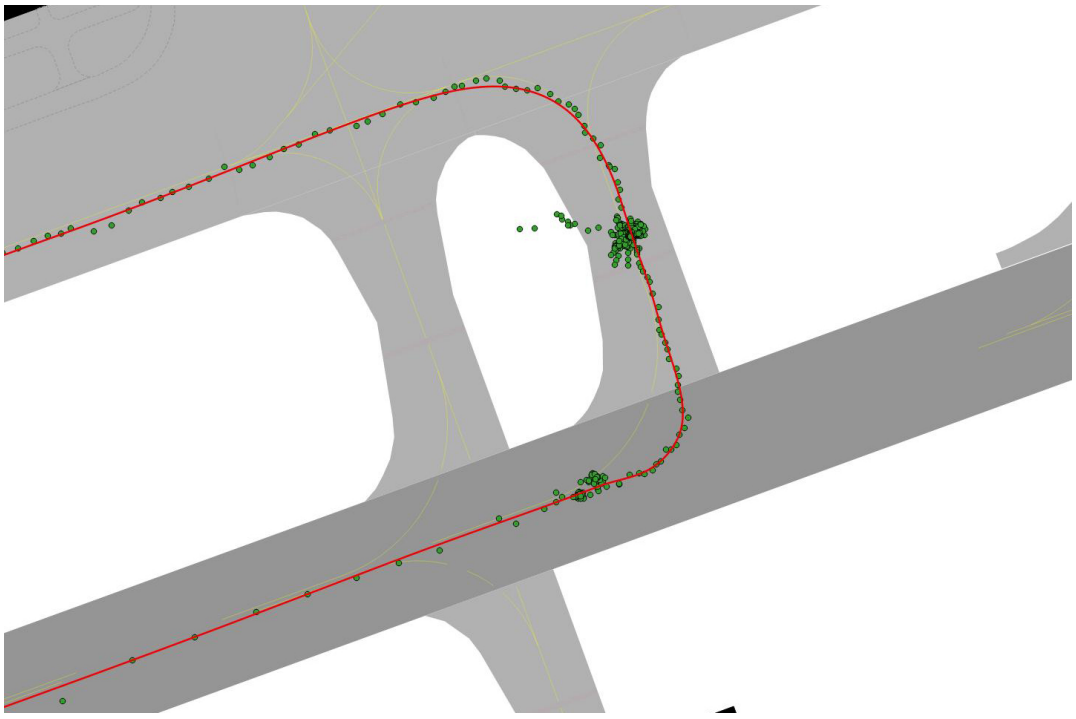


(b) Robust smoothing algorithm presented in this thesis without standstill boundary condition.

Figure 9.2: The difference between robust and ordinary least-squares fitting is obvious. Even with drastically smaller smoothing penalty the fitted curve is less influenced by outliers.

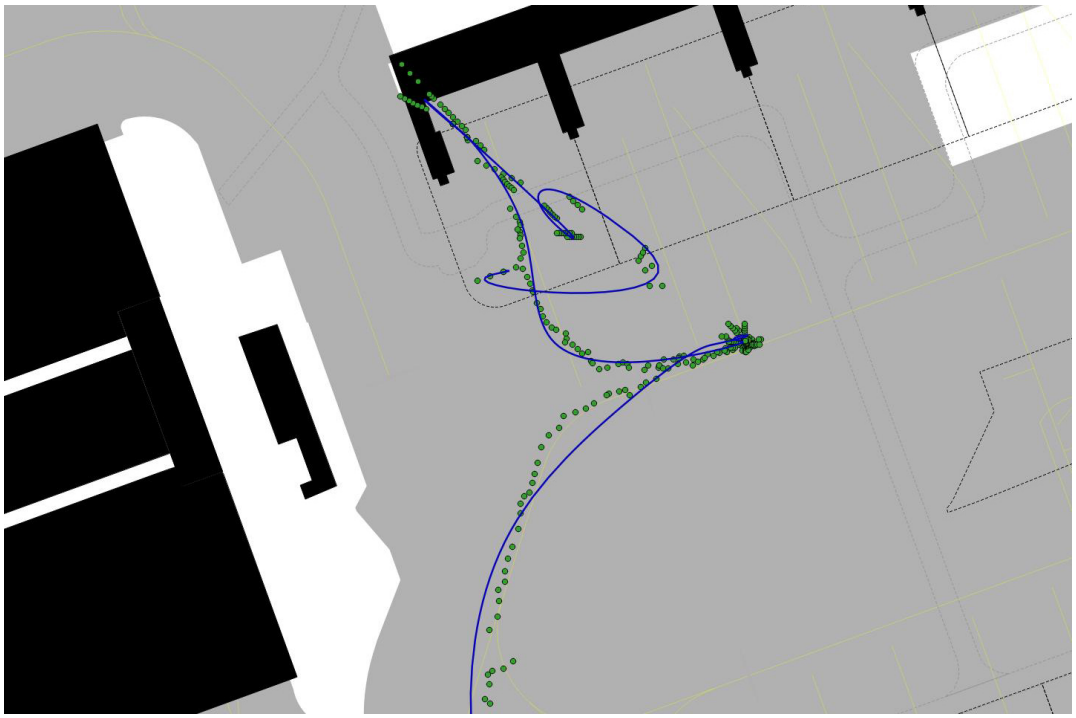


(a) Robust approximation without boundary conditions.

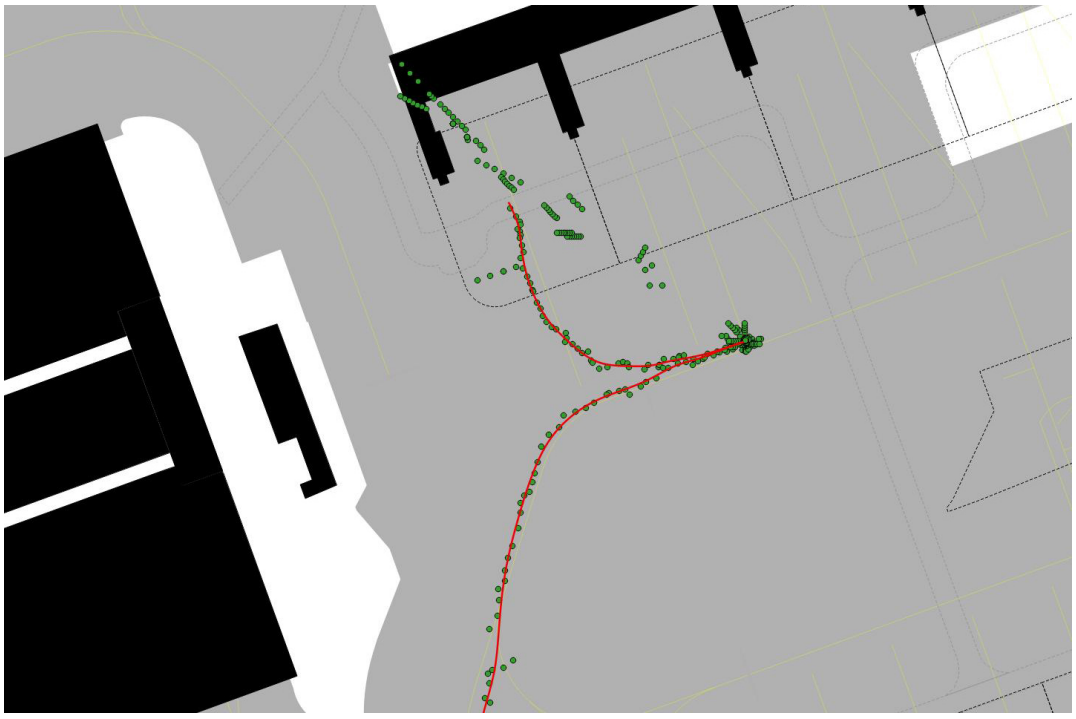


(b) Robust approximation with standstill boundary conditions

Figure 9.3: Adding standstill boundary conditions to the approximation guarantees the approximation to be constant within the standstill interval. The resulting fit closely resembles the actual behaviour of the aircraft.



(a) Robust approximation without boundary conditions.



(b) Robust approximation with standstill boundary conditions added.

Figure 9.4: The standstill boundary conditions do not only enforce physically correct behaviour but also reduce the effect of outliers throughout the standstill. The pushback process now is clearly detectable in the approximation.

the problem is split up as described this time decreases to 0.05 seconds. Additionally determining the standstill intervals using DBSCAN and the refinement algorithm described in chapter 7.4 increases the average computation time to 0.1 seconds.

In summary we have shown that the derived algorithms obtain good results both in the absence and presence of noise and outliers. The performance of the algorithms is significantly better than Fraport's old least squares approach while at the same time more robust against outliers. We may conclude that the design goals of this thesis have been achieved.

Nevertheless, there are possibilities for further research that shall not be left unmentioned. We have seen that the quality of the approximation is strongly influenced by the choice of the smoothing parameter. In chapter 8.3 we suggested that this parameter could be selected based on the region the points were recorded in. First tests of this approach yielded good results but this has not been verified on a large scale.

One important aspect that has not been covered in this thesis is self-verification of the algorithm as well as recovery in case of wrong approximation results. Is it possible for the approximation algorithm to estimate its own approximation quality and automatically point out approximation problems?

One way to do this would require defining a measure of quality and detecting deviations from the usual values. This would not only allow to identify regions of bad approximation but could also be used as a test on the integrity of the provided data. This approach even could be translated to near-real time data: In the knowledge that the position information the A-SMGCS provides are used increasingly for traffic control it would be valuable to identify problems with the data sources automatically and send a warning to the responsible apron controllers.

Another way to detect at least obvious outliers and approximation errors would be to compare the found velocities to common velocities in the same region. In this case velocities that are too high could indicate approximation problems and justify increasing the smoothing parameter in this region.

Some data source also transmit an estimate on the velocity of the aircraft. This estimate could be used as additional data source in the approximation, i.e. we could add auxiliary summands $\rho^*(v_i - f'(t_i))$ to further increase the approximation accuracy. The approximation also could be further restricted by a boundary condition limiting the maximal velocity, which is $\|f'(x)\|_2^2 < c$. Both approaches promise a further robustness against outliers.

Finally, this thesis only focuses on the approximation of the data. It is of course necessary to use the found spline approximations later on to commence actual analysis tasks. This requires the development of various algorithms for the processing of the found splines. In this regard Fraport already created a broad toolkit of possible procedures that work on polygonal chains. Those can easily be obtained from the spline approximations. In terms of performance and accuracy it is desired to replace the mentioned algorithms successively with similar ones working directly on splines.

From a mathematical point of view the central result of this thesis certainly is the generalization of the IRLS algorithm to arbitrary optimization procedures. While this method has been used previously, this seems to be the first work actually proving the convergence of the method in such a general setting. Furthermore the idea how to solve the quadratic smoothing program under a simple, quadratic boundary condition can serve as a blueprint to derive solution methods for similarly structured problems.

Appendices

A Numerical foundations

Throughout the following some fundamental numerical algorithms shall be presented. While neither new nor spectacular, they build the foundation for some more sophisticated algorithms presented before. All procedures will be presented in a compact form and without further proofs.

A.1 Quadrature Rules

Typical numerical integration tries to approximate an integral by a weighted sum of function values. Such a *quadrature rule* is given by

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(c_i).$$

The values $w_i \in \mathbb{R}$ are the weights multiplied to the function values at knots $c_i \in \mathbb{R}$.

If for some weighting function w a polynomial q of degree m can be exactly integrated, e.g. the equation

$$\int_{-1}^{1} w(x) f(x)\,\mathrm{d}x = \sum_{i=1}^{n} w_i f(x_i)$$

holds. The quadrature rule is said to exact up to degree m. Finally, a quadrature rule that is exact up to degree $2n-1$ is said to be *Gaussian*.

For the purpose of this thesis, only the weighting function $w \equiv 1$ is of interest. A Gaussian quadrature rule for that case can be given using the Legendre knots and weights. For $n = 1, \ldots, 4$ those weights are given in Table A.1.

When integration is to be done on an arbitrary interval $[a, b]$ rather than $[-1, 1]$, variable transformation can be used to obtain the desired results:

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}x + \frac{a+b}{2}\right)\mathrm{d}x \approx \frac{b-a}{2} \sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}c_i + \frac{a+b}{2}\right).$$

Further details on numerical integration can be found in [20].

	c_i	w_i
$n = 1$	0	2
$n = 2$	−0.577350269189626	1
	0.577350269189626	1
$n = 3$	−0.774596669241483	0.555555555555554
	0	0.888888888888889
	0.774596669241483	0.555555555555554
$n = 4$	−0.861136311594053	0.347854845137454
	−0.339981043584856	0.652145154862546
	0.339981043584856	0.652145154862546
	0.861136311594053	0.347854845137454

Table A.1: The Legendre knots and weights used for Gaussian quadrature on the interval $[-1, 1]$ for $n = 1, \ldots, 4$.

A.2 Computations with polynomials

Chapter 7.3 requires the numerical computations with polynomials, primarily their multiplication and evaluation.

We may save a polynomial by saving a vector of its coefficients:

$$[p_0, \ldots, p_n]^\mathsf{T}$$

correlates to

$$p(x) := \sum_{i=0}^{n} p_i x^i.$$

Addition and subtraction then can simply be defined as straightforward vector addition. If both polynomials were of different degree, the vector of lesser degree just would have to be filled up with zeros accordingly.

For our purpose it is sufficient to define polynomial multiplication in the trivial way, thus for two polynomials q and q the coefficients of $r = pq$ are

$$r_i = \sum_{j=0}^{i} p_j q_{i-j}.$$

Zeros are inserted whenever the indices do not exist. There are more efficient ways for polynomial multiplication available, for example by Karatsuba and Ofman or based on the Fast Fourier Transform. See [32] for an in-depth discussion on this topic.

Evaluating polynomials may be achieved using the Horner scheme: Let p be a polynomial of degree n. Then

$$\begin{aligned} b_n &= p_n \\ b_i &= x b_{i+1} + p_i, \qquad i = n-1, \ldots, 0 \end{aligned}$$

and $p(x) = b_0$ [33, pp. 280 – 282].

A.3 Root finding

Finding the roots of real valued functions is a well investigated concept with plenty of different methods available. Purely for reference we will include a few different algorithms that may be combined to find zeros for arbitrary functions $f \in C^2(\mathbb{R})$. More information on one-dimensional optimization can be found in any textbook on numerical analysis, for example in [56], [60].

One of the most elementary methods is the bisection algorithm.

Algorithm A.1: Bisection

Input: Function f.
Confidence interval $[a, b]$ with $f(a)f(b) < 0$.
Error tolerance ε.

Result: Value t with $|f(t)| \leq \varepsilon$.

```
repeat
    t ← (a+b)/2
    if f(a)f(t) > 0 then
        a ← t
    else
        b ← t
    end if
until |f(t)| > ε
```

Due to the fact that the condition $f(a)f(b) < 0$ secures a root within the given confidence interval the bisection method is guaranteed to converge. The disadvantage of the method, however, is that it only offers linear convergence.

Newton's method on the other hand assures quadratic convergence as long as the starting value is sufficiently close to the actual root but does not converge in every situation. Both methods can be combined to gain an efficient and secure algorithm:

Algorithm A.2: Newton-Bisection-Hybrid

Input: Function f and its derivative f'.
Confidence interval $[a, b]$ with $f(a)f(b) < 0$.
Error tolerance ε.
Maximum number of successive Newton iterations $c \in \mathbb{N}$.

Result: Value t with $|f(t)| \leq \varepsilon$.

```
t ← (a+b)/2
i ← 0
repeat
    if |f'(t)| < ε or t ∉ [a, b] or i > C then          ▷ One step bisection
        t ← (a+b)/2
        i ← 0
    else                                                ▷ One step Newton's method
        t ← t − f(t)/f'(t)
        i ← i + 1
        if f(a)f(t) > 0 then                            ▷ Update confidence interval
            a ← t
        else
            b ← t
        end if
    end if
until |f(t)| > ε
```

List of algorithms

Symbols and notation

$0_{j,k}$ A zero matrix in $R^{j \times k}$.

$\lceil \cdot \rceil$ The ceiling operator that returns the smallest integer greater than the argument.

$\lfloor \cdot \rfloor$ The floor operator that returns the largest integer less than the argument.

1_k A $k \times k$ identity matrix.

$\|\cdot\|_1$ The l_1-norm, i.e. $\|x\|_1 = \sum_{i=0}^{l-1} |x_i|$ with $x = [x_0, \ldots, x_l]$.

$\|\cdot\|_2$ The euclidean norm.

d_i The i-th control point of a spline curve.

$\mathbf{d}$ The matrix $[d_0, \ldots, d_{n-1}]^\mathsf{T}$ of control points.

$\mathbf{d}_\beta$ β-column of $\mathbf{d}$ with $\beta \in \{x, y, z\}$, i.e. $\beta = [\mathbf{d}_x, \mathbf{d}_y, \mathbf{d}_z]$.

$\mathrm{diag}(\cdots)$ A diagonal matrix with the functions arguments on the diagonal.

e_i i-th unit vector.

f A function $f \colon \mathbb{R} \to \mathbb{R}^l$. In most cases used as function mapping time to a vehicles position.

f_β β-component of f with $\beta \in \{x, y, z\}$, i.e. $f = [f_x, f_y, f_z]$.

$f_i(x)$ Shorthand for $f(t_i, x)$.

G_k The k-th derivative matrix of a spline.

l The dimension of an object. Usually $l \in \{2, 3\}$.

m The number of data points available.

n The number of control points of a spline curve.

$N_{i,q}(t \mid \mathcal{T})$ The i-th B-spline of degree q to the knot sequence $\mathcal{T}$.

$N_{i,q}(t)$ Shorthand for $N_{i,q}(t \mid \mathcal{T})$ whenever the knot sequence $\mathcal{T}$ is clear from the context.

$N_q(t)$ The vector $[N_{0,q}(t), \ldots, N_{n-1,q}(t)]^\mathsf{T}$.

$N_q(\mathbf{T})$ The collocation matrix $[N_q(t_0), \ldots, N_q(t_{m-1})]^\mathsf{T}$ for $\mathbf{T} = [t_0, \ldots, t_{m-1}]^\mathsf{T}$.

P_i $P_i = f(t_i)$. Furthermore $P_i = [P_{x,i}, P_{y,i}, P_{z,i}]^\mathsf{T}$.

$\mathbf{P}$ The matrix $\mathbf{P} = [P_0, \ldots, P_{m-1}]^\mathsf{T}$.

$\mathbf{P}_\beta$ The vector $\mathbf{P} = [P_{\beta,0}, \ldots, P_{\beta,m-1}]^\mathsf{T}$ with $\beta \in \{x, y, z\}$.

q The degree of a spline. $q + 1$ is called the order of a spline.

$r_i(x)$ The residual $y_i - f_i(x)$.

ρ A measurable, positive and convex function that is used as distance measure for M-estimators.

ρ_c The Huber loss function, a special M-estimate. See (4.15).

ρ^* Multidimensional expansion of ρ, with $\rho^*(x) := \sum_{i=0}^{l} \rho(x_i)$.

S_q^l The linear space $\left\{ \sum_{i=0}^{n-1} d_i N_{i,q}(t \mid \mathcal{T}) \;\middle|\; d_i \in \mathbb{R}^l \right\}$.

$\mathbb{S}_q(\mathcal{T})$ The vector space that is spanned by the B-splines of degree q with respect to the knot sequence $\mathcal{T}$.

$\mathcal{T}$ Knot sequence $\mathcal{T} = [\tau_0, \ldots, \tau_{n+q}]$ with $\tau_i \leq \tau_{i+1}$.

τ_i The i-th knot of the knot sequence $\mathcal{T}$.

$\mathcal{T}^{(k)}$ A modification of the knot sequence $\mathcal{T}$ that has $q + 1 - k$-fold boundary knots.

t_i Timestamp $t_i \in \mathbb{R}$ the function f has been sampled at.

$\mathbf{T}$ The vector $\mathbf{T} := [t_0, \ldots, t_{m-1}]$.

W Diagonal matrix of weights, $W = \mathrm{diag}(w_0, \ldots, w_{m-1})$.

w_i Weights for data points within the approximation problem, see eq. (4.3).

$\sqrt{W}$ Diagonal matrix of the square root of weights, $W = \mathrm{diag}(\sqrt{w_0}, \ldots, \sqrt{w_{m-1}})$.

$\mathbf{Y}$ The vector $\mathbf{Y} := [y_0, \ldots, y_{m-1}]$ of data points to be approximated.

Bibliography

[1] C. Adcock and N. Meade, "A comparison of two LP solvers and a new IRLS algorithm for l_1 estimation," in *L_1-statistical procedures and related topics*, Y. Dodge, Ed., vol. 3, IMS Lecture Notes, 1997, pp. 119–132.

[2] *Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Manual*, Doc. 9830. International Civil Aviation Organization, 2004.

[3] I. Barrodale and F. D. K. Roberts, "An efficient algorithm for discrete l_1 linear approximation with linear constraints," *SIAM Journal on Numerical Analysis*, vol. 15, no. 3, pp. 603–611, 1978.

[4] I. Barrodale and F. D. K. Roberts, "An improved algorithm for discrete l_1 linear approximation," *SIAM Journal on Numerical Analysis*, vol. 10, no. 5, pp. 839–848, 1973.

[5] I. Barrodale and F. D. K. Roberts, "Applications of mathematical programming to l_p approximation," *Nonlinear Programming*, pp. 447–464, 1970.

[6] H. E. Bell, "Gershgorin's theorem and the zeros of polynomials," *The American Mathematical Monthly*, vol. 72, no. 3, pp. 292–295, 1965.

[7] Å. Björck, *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, 1996.

[8] P. Bloomfield and W. L. Steiger, *Least Absolute Deviations. Theory, Applications and Algorithms*, ser. Progress in Probability and Statistics. Birkhäuser, 1983, vol. 6.

[9] P. Bloomfiled and W. L. Steiger, "Least absolute deviations curve-fitting," *SIAM Journal on Scientific and Statistical Computing*, vol. 1, pp. 290–301, 1980.

[10] C. de Boor, *A Practical Guide to Splines*, 2nd ed., ser. Applied Mathematical Sciences 27. Springer, 2001.

[11] C. de Boor, "Good approximation by splines with variable knots. II," in *Conference on the Numerical Solution of Differential Equations*, G. Watson, Ed., ser. Lecture Notes in Mathematics, vol. 363, Springer, 1974, pp. 12–20.

[12] C. de Boor and G. J. Fix, "Spline approximation by quasiinterpolants," *Journal of Approximation Theory*, vol. 8, no. 1, pp. 19–45, 1973.

[13] K. W. Bosworth and U. Lall, "An l_1 smoothing spline algorithm with cross validation," *Numerical Algorithms*, vol. 5, no. 8, pp. 407–417, 1993.

[14] I. Bronshtein, K. Semendyayev, G. Musiol, and H. Mühlig, *Handbook of Mathematics*. Springer, 2007.

[15] P. L. Butzer, M. Schmidt, and E. L. Stark, "Observations on the history of central B-splines," *Archive for History of Exact Sciences*, vol. 39, pp. 137–156, 2 1988.

[16] R. H. Byrd and D. A. Pyne, "Convergence of iteratively reweighted least squares algorithm for robust regression," Department of Mathematical Sciences, The John Hopkins University, Tech. Rep. 313, 1979.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.

[18] D. D. Cox, "Asymptotics for M-type smoothing splines," *The Annals of Statistics*, vol. 11, no. 2, pp. 530–551, 1983.

[19] P. Craven and G. Wahba, "Smoothing noisy data with spline functions," *Numerische Mathematik*, vol. 31, pp. 377–403, 4 1978.

[20] P. Davis and P. Rabinowitz, *Methods of Numerical Integration*, 2nd ed., ser. Dover Books on Mathematics Series. Dover Publications, 2007.

[21] R. DeVore and G. Lorentz, *Constructive Approximation*, ser. Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen. Springer, 1993.

[22] N. Draper and H. Smith, *Applied regression analysis*, 3rd ed., ser. Wiley Series in Probability and Statistics. Wiley, 1998.

[23] R. Dutter, "Robust regression: Different approaches to numerical solution and algorithms," Fachgruppe für Statistik, Eidgenössische Technische Hochschule, Zürich, Tech. Rep., 1975.

[24] R. Dutter and P. J. Huber, "Numerical methods for the nonlinear robust regression problem," *Journal of Statistical Computation and Simulation*, vol. 13, no. 2, pp. 79–113, 1981.

[25] O. Edlund, H. Ekblom, and K. Madsen, "Algorithms for non-linear M-estimation," *Computational Statistics*, vol. 12, pp. 373–383, 1997.

[26] H. Ekblom and K. Madsen, "Algorithms for non-linear Huber estimation," *BIT Numerical Mathematics*, vol. 29, no. 1, pp. 60–76, 1989.

[27] ERA a.s. (2014). Multilateration - executive reference guide, [Online]. Available: `http://www.multilateration.com` (visited on 12/05/2014).

[28] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Procedings of the Second International Conference on Knowledge Discovery and Data Mining*, E. Simoudis, J. Han, and U. Fayyad, Eds., AAAI, 1996, pp. 226–231.

[29] G. Farin, *Curves and Surfaces for CAGD*. Academic Press, 2002.

[30] Flightradar24 AB. (2015). Flightradar24, [Online]. Available: `http://www.flightradar24.com` (visited on 12/05/2014).

[31] Fraport AG, "S.O.D.A.," Software toolbox for analyzing A-SMGCS data, 2014.

[32] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge University Press, 2003, ISBN: 9780521826464.

[33] W. Gautschi, *Numerical Analysis*, 2nd ed. Birkhäuser, 2012.

[34] G. H. Golub and V. Pereyra, "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate," *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 413–432, 1973.

[35] G. Golub and V. Pereyra, "Separable nonlinear least squares: The variable projection method and its applications," *Inverse Problems*, vol. 19, no. 2, R1–R26, 2003.

[36] G. Golub and C. Van Loan, *Matrix Computations*, 4th ed. Johns Hopkins University Press, 2012.

[37] P. Hansen, V. Pereyra, and G. Scherer, *Least Squares Data Fitting with Applications*. Johns Hopkins University Press, 2012.

[38] M. Hazewinkel, *Encyclopaedia of Mathematics*. Springer, 1989, vol. 4.

[39] M. Hirz, W. Dietrich, A. Gfrerrer, and J. Lang, *Integrated Computer-Aided Design in Automotive Development: Development Processes, Geometric Fundamentals, Methods of CAD, Knowledge-Based Engineering Data Management*. Springer, 2013.

[40] K. Höllig and J. Hörner, *Approximation and Modeling with B-Splines*. Society for Industrial and Applied Mathematics, 2014.

[41] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[42] P. J. Huber, "Robust smoothing," in *Robustness in Statistics: Proceedings of a Workshop*, R. Launer and G. Wilkinson, Eds., Academic Press, 1979, pp. 33–48.

[43] P. J. Huber, "The behavior of maximum likelihood estimates under nonstandard conditions," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, University of California Press, 1967, pp. 221–233.

[44] P. J. Huber and E. M. Ronchetti, *Robust Statistics*, 2nd ed., ser. Wiley Series in Probability and Statistics. Wiley, 2009.

[45] J. J. E. Dennis, "Non-linear least squares and equations," in *State of the Art in Numerical Analysis*, D. A. H. Jacobs, Ed., Academic Press, 1977, pp. 269–306.

[46] L. Kaufman, "A variable projection method for solving separable nonlinear least squares problems," *BIT Numerical Mathematics*, vol. 15, no. 1, pp. 49–57, 1975.

[47] J. S. Lee and D. D. Cox, "Robust smoothing: Smoothing parameter selection and applications to fluorescence spectroscopy," *Computational Statistics & Data Analysis*, vol. 54, no. 12, pp. 3131–3143, 2010.

[48] T. C. M. Lee and H.-S. Oh, "Robust penalized regression spline fitting with application to additive mixed modeling," *Computational Statistics*, vol. 22, no. 1, pp. 159–171, 2007.

[49] D. Lei, I. Anderson, and M. Cox, *A Robust Algorithm for Least Absolute Deviations Curve Fitting*. Defense Technical Information Center, 2001.

[50] R. V. Lenth, "Robust splines," *Communications in Statistics - Theory and Methods*, vol. 6, no. 9, pp. 847–854, 1977.

[51] Z. Liu and W. Guo, "Data driven adaptive spline smoothing," *Statistica Sinica*, pp. 1143–1163, 2010.

[52] J. McNamee, *Numerical Methods for Roots of Polynomials - Part 1*. Elsevier Science, 2007.

[53] J. McNamee and V. Pan, *Numerical Methods for Roots of Polynomials - Part 2*. Elsevier Science, 2013.

[54] H. Mensen, *Planung, Anlage und Betrieb von Flugplätzen*. Springer, 2007.

[55] E. Mikhail and F. Ackermann, *Observations and least squares*. University Press of America, 1982.

[56] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[57] M. Osborne, "Separable least squares, variable projection, and the gauss-newton algorithm," *Electronic Transactions on Numerical Analysis*, vol. 28, no. 2, pp. 1–15, 2007.

[58] L. Piegl and W. Tiller, *The NURBS Book*. Springer, 1995.

[59] A. Pintore, P. Speckman, and C. C. Holmes, "Spatially adaptive smoothing splines," *Biometrika*, vol. 93, no. 1, pp. 113–125, 2006.

[60] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[61] A. Ruhe and P. Wedin, "Algorithms for separable nonlinear least squares problems," *SIAM Review*, vol. 22, no. 3, pp. 318–337, 1980.

[62] E. J. Schlossmacher, "An iterative technique for absolute deviations curve fitting," *Journal of the American Statistical Association*, vol. 68, no. 344, pp. 857–859, 1973.

[63] G. Seber and C. Wild, *Nonlinear Regression*, ser. Wiley Series in Probability and Statistics. Wiley, 2003.

[64] C. B. Storlie, H. D. Bondell, and B. J. Reich, "A locally adaptive penalty for estimation of functions with varying roughness," *Journal of Computational and Graphical Statistics*, vol. 19, no. 3, pp. 569–589, 2010.

[65] M. Tepper and G. Sapiro, "L1 splines for robust, simple, and fast smoothing of grid data," 2012. arXiv: `1208.2292v2 [cs.NA]`.

[66] F. Utreras, "On computing robust splines and applications," *SIAM Journal on Scientific and Statistical Computing*, vol. 2, no. 2, pp. 153–163, 1981.

[67] C. Vogel, *Computational Methods for Inverse Problems*, ser. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 2002.

[68] H. M. Wagner, "Linear programming techniques for regression analysis," *Journal of the American Statistical Association*, vol. 54, no. 285, pp. 206–212, 1959.

[69] G. Wahba, "A comparison of GCV and GML for choosing the smoothing parameter in the generalized spline smoothing problem," *Annals of Statistics*, vol. 13, pp. 1378–1402, 1985 1985.

[70] G. Wahba, *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

[71] R. Wolke and H. Schwetlick, "Iteratively reweighted least squares: Algorithms, convergence analysis, and numerical comparisons," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 5, pp. 907–921, 1988.

[72] R. Wolke, "Iteratively reweighted least squares: A comparison of several single step algorithms for linear models," *BIT Numerical Mathematics*, vol. 32, no. 3, pp. 506–524, 1992.