1 Introduction

The ongoing digitalization of complex, large-scale industrial plants is leading to a massive increase of process data. This data can be used to enhance the overall understanding of the characterizing physical process inside the plant. Modern observer and control concepts are used to enhance the efficiency and quality of the plant. They use mathematical models of the ongoing process. The exact physical description of the relevant quantities as mathematical model is nevertheless often not feasible because of the complexity of the process as well as computational and measurement limitations.

Data driven approaches are state-of-the-art in many fields, including e.g. image and speech recognition. The usage of data driven methods, e.g. artificial neural networks, parametric models, etc., to model process quantities for which measurements are expensive or not practical, gains more and more influence and acceptance in the field of process control and optimization. The application of the specific algorithm depends on issues like data quality, interpretability of the model and computational efficiency.

In most process optimization tasks massive amounts of domain specific knowledge in form of physical theories and a priori knowledge is available. The combination of the use of domain knowledge and data driven modeling techniques is called hybrid modeling or grey-box modeling. It lies between the two modeling extrema of white-box models, which are derived from first principles and physical models and black-box models, which are derived from data only[**ashby1961introduction**]. The incorporation of this knowledge in state-of-the-art data driven approaches is not trivial and not solved for some algorithms. Nevertheless, its inclusion should improve the interpretability, which itself is of importance in the context of the emerging field of explainable artificial intelligence (XAI). XAI refers to modeling approaches and techniques in which the main goal is that the resulting model is understandable by humans[**dovsilovic2018explainable**].

In this thesis, we are going to develop an algorithm for efficient, static, multi-dimensional function approximation using a priori domain knowledge. The algorithm is based on structured additive regression using splines[**fahrmeir2007regression**]. We use user-defined constraints to include the a priori domain knowledge in the fitting process[**hofner2011monotonicity**]. It should produce interpretable and efficient models based on the given domain knowledge. The incorporation of domain specific knowledge should improve the model quality and robustness as well as interpolation and extrapolation behavior in situations where the measured data is sparse and/or noisy. Further, we are going to evaluate the algorithm using noisy samples from artificial test functions with known behavior as well as real world data collected in a heat treatment process.

1.1 Related Work

We will now discuss some of the most used data-driven algorithms. The discussion includes the following model approaches:

- Parametric models: Linear and polynomial regression
- Non-parametric models: Basis function models
- Artificial neural networks
- Look-up tables

and focuses on the interpretability, computational efficiency and the ability to include domain knowledge of the individual modeling approaches. The list given above is not intended to be complete.

The common starting point for the different data-driven modeling approaches is that we have some data $\{x_1^{(i)}, \dots, x_q^{(i)}; y^{(i)}\},\ i = 1, \dots, n$. We restrict the following discussion to the single-input setting, i.e. $\{x^{(i)}, y^{(i)}\},\ i = 1, \dots, n$. The generalization to multiple input dimensions is given in the respective literature. We then use the given data to estimate a model function $f(x)$ which is then used to predict the response or output variable y, i.e.

$$y = f(x). \tag{1.1}$$

Therefore we are in the setting of supervised learning.

1.1.1 Parametric Models

According to Nelles, parametric models are defined as models that can describe the true process behavior using a finite number of parameters[**nelles2013nonlinear**]. An example is given by the linear regression model for one input variable x as

$$y = f(x) = \beta_0 + \beta_1 x. \tag{1.2}$$

Both parameters β_0 and β_1 allow for a direct interpretation as β_0 is the intercept, i.e. the output for the input $x = 0$, and β_1 is the slope, i.e. the constant defining the relationship between the increase of the output y with respect to the increase of the input x. The interpretability of linear regression models is therefore very high.

Linear regression models are widely used and part of standard software tools. Their parameters can be efficiently computed using the least squares algorithm. One major drawback is that they can only recover linear relationships between input and output variables. They are therefore quite restrictive and do not allow the incorporation of a priori domain knowledge except being increasing or decreasing by the sign of the slope β_1.

An extension of the linear regression model is given by polynomial regression. Here, we try to model the output data y using a polynomial of degree p, i.e.

$$y = f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p. \tag{1.3}$$

Polynomial regression introduces more flexibility in the fitting process, since the restriction of linear relationship is relaxed to a polynomial relationship of degree p. As for linear models, the interpretability of the parameters is given. We can use the least squares algorithm for parameter estimation. The incorporation of some a priori domain knowledge is possible, e.g. as the degree of the polynomial regression model. The major problem of polynomial regression is that the model function becomes quite wiggly for high polynomial degrees p.

Linear and polynomial regression models are so-called global models. Their parameters act on the complete input space. This property makes the incorporation of specific a priori domain knowledge, e.g. unimodal behavior, difficult and in most cases nearly impossible for parametric models.

1.1.2 Non-parametric Models

Nelles defines non-parametric models as models which require an infinite number of parameters to describe a process exactly[**nelles2013nonlinear**]. In almost all practical applications this infinite series is approximated by a finite number of parameters using the basis function approach given by

$$y = f(x) = \sum_{i=1}^{M} \theta_i^{(l)} \Phi_i(x, \theta_i^{(nl)}) \tag{1.4}$$

with the linear parameters $\theta_i^{(l)}$, the basis functions $\Phi_i(.)$, the input variable x and the non-linear parameters $\theta_i^{(nl)}$. The output y is therefore given by a linear combination of M basis functions $\Phi_i(.)$. To model a non-linear relationship between y and x, the basis functions $\Phi(.)$ need to be non-linear. Commonly used basis functions are e.g. the *hat function*, the *Gaussian*, *splines* or the *hinge function*.

A commonly used algorithm utilizing the basis function approach is called Multivariate Adaptive Regression Splines (MARS) [**friedman1991multivariate**]. MARS approximates data, as example again for a single input dimension, using the following model

$$y = \sum_{i=1}^{M} \theta_i \Phi_i(x) \tag{1.5}$$

using constant parameters θ_i. The basis functions $\Phi_i(.)$ are one of the following three alternatives:

1. $\Phi_i(x) = 1$, representing the intercept.

2. $\Phi_i(x) = \max(0, x - c_i)$ or $= \max(0, c_i - x)$, representing the *hinge function* h_i using the constant value c_i.

3. $\Phi_i(x) = h_i h_j$, representing a product of two *hinge functions.*

MARS fits the model using a recursive splitting approach. More information can be found in [**friedman1991multivariate**] and [**friedman2001elements**]. MARS models are more flexible compared to the parametric linear and polynomial regression models. As only hinge functions and products of hinge functions are used, MARS models are efficient and in general simple to understand and interpret. To our knowledge, there is currently no possibility to include a priori domain knowledge in the fitting process when using MARS.

Another widely used methods using basis functions is the use of *splines.* Splines are defined as piece-wise polynomials on a sequence of knots. Further information can be found in Section **??** and [**deBoor1978practicalGuideToSplines**]. Using k splines to model some data, we obtain the following model formulation

$$y = f(x) = \sum_{i=1}^{k} \beta_i b_i(x) \tag{1.6}$$

with the spline basis functions b_i and the parameters β_i. The parameters can be calculated used the least squares algorithm. The usage of splines allows a lot of flexibility and computational efficiency. A priori domain knowledge can be incorporated using the penalized least squares algorithm with a sophisticated choice of mapping and weighting matrices, see **??**, and e.g. [**hofner2011monotonicity**] or [**bollaerts2006simple**].

The basis function approach in (**??**) may be extended by changing the parameters $\theta_i^{(l)}$ to more complex forms. An example for this is the so-called local linear neuro-fuzzy model, for which each parameter $\theta_i^{(i)}$ is changed to be a *local linear model* and each basis function $\Phi_i(.)$ is then called *validity function* determining the region of validity of the local linear model [**nelles2013nonlinear**]. The validity functions are normalized for any model input x, i.e.

$$\sum_{i=1}^{M} \Phi_i(x) = 1. \tag{1.7}$$

and typically chosen to be *Gaussian* functions, i.e.

$$\Phi_i(x) = a_i \exp\left(\frac{(x-\mu_i)^2}{\sigma_i^2}\right) \tag{1.8}$$

with the normalization constant a_i and the parameters μ_i and σ_i determining the location and scale of the Gaussian function. The output of the local linear neuro-fuzzy model using M local linear models is then given by

$$y = \sum_{i=1}^{M} (\beta_{i0} + \beta_{i1} x_1) \Phi_i(x). \tag{1.9}$$

The first term in the summation are the *local linear models*. The parameters β_{ij} for $i = 1, \ldots, M$ and $j = 0, 1$ as well as the parameters μ_i and σ_i from the validity functions Φ_i need to be optimized. This is done using the LOLIMOT algorithm. Further information is given in [**nelles2013nonlinear**].

Local linear models as extension of linear models possess more flexibility with regards to non-linear relationships in the data. They can also be efficiently evaluated after the iterative training process. The interpretability is high since each local linear model contributes to the prediction according to its validity function. The ability to include a priori domain knowledge in the fitting process is currently not available.

1.1.3 Artificial Neural Networks

Artificial neural networks are currently the state-of-the-art solution method for many problems ranging from computer vision over time-series prediction to regression tasks. They are constructed as coarse model of the human brain, consisting of neurons which are connected by some weights. These connections are adapted in the learning process using an algorithm called "backpropagation". They utilize a high number of parameters to model hidden, high-dimensional relationships in the data. Further information can be found in standard textbook about neural networks, e.g. [**bishop2006patternRecognition**] or [**goodfellow2016deep**].

In terms of modeling flexibility, artificial neural networks of sufficient size are proven to be able to represent a wide variety of functions by so-called universal approximation theorems, cf. [**cybenko1989approximation**] and [**hornik1991approximation**]. The computational complexity of a neural network depends on its size, aka. the number of parameters. Large networks need many training samples to generate sufficiently accurate predictions. Artificial neural networks are an example of a black-box model. The inclusion of a priori domain knowledge into the learning process of neural networks is possible for specific types of knowledge using the concepts of hints, see [**abu1990learning**] and [**sill1997monotonicity**].

1.1.4 Look-up Tables

A look-up table is an array of values, which allows to replace computational expensive computations with inexpensive array indexing operations. The values in the look-up table are most often computed and stored beforehand. To gain higher resolution, interpolation techniques such as linear or quadratic interpolation may be applied to look-up tables.

Look-up tables are a standard tool in many fields. They are extremely efficient in terms of computation time. One problem that occurs is the exponential increase in size with the number of dimensions for the look-up table. As example, a 2×2-table needs to save 4 values, while a $2 \times 2 \times 2$ table already needs 8 values without gaining additional accuracy. Another problem is that the values in the look-up table may come from complex, computational or physical models.

Lattice regression tackles this problems by jointly estimating all lookup-table values by minimizing the regularized interpolation error on training data [**garcia2009lattice**]. They state that using ensembles of lookup-tables which combine several *tiny* lattices enables linear scaling in the number of input dimension even for high dimensions [**fard2016fast**]. They further state that lattice regression may be used to incorporate a priori domain knowledge like monotonicity, shape or unimodality into the fitting process, see [**gupta2016monotonic**] or [**you2017deep**].

1.2 Outline

The base of this thesis is a literature study about function fitting using a priori domain knowledge focusing on non-parametric techniques and neural networks. We decided to use structured additive regression [**fahrmeir2007regression**] utilizing B-splines and tensor product splines as non-linear basic functions. This approach enables flexible, multi-dimensional function fitting. We further expanded this method by applying a priori domain knowledge through the use of user-defined constraints. These constraints consist of mapping matrices determined by the type of constraint, and weighting matrices, determining whether the constraint is active, see [**hofner2011monotonicity**] and [**bollaerts2006simple**].

We are able to incorporate the following a priori domain knowledge:

- Monotonicity, i.e. $f'(x) \geq 0$ or $f'(x) \leq 0$
- Curvature, i.e. $f''(x) \geq 0$ or $f''(x) \leq 0$
- Unimodality, i.e.
  $f'(x) > 0, \ x < m$ and $f'(x) < 0, \ x > m$ for $m = \arg\max_x f(x)$
- Boundedness, i.e. $f(x) \geq M$ or $f(x) \leq M$ for some value M
- Jamming, i.e. $f(x^{(p)}) \approx y^{(p)}$ for some point p with high fidelity

The thesis is divided into 5 chapters: Chapter 2 provides an overview of the fundamental mathematical concepts used. We focus on the description of linear models, splines and the topic of structured additive regression. In chapter 3, we develop the algorithm using the concepts given in chapter 2. In chapter 4, we test the algorithm using different artificial functions and a priori domain knowledge as well as real-world data. Chapter 5 gives a summary and outlines future, possible work.

2 Practical Considerations

In Chapter **??**, we discussed the theoretical basis for shape-constraint P-splines and their ability to incorporate a priori domain knowledge by choice of the constraint term described by the mapping matrix $\mathbf{D}_c$ and the weighting matrix $\mathbf{V}_c$. We will now consider the practical application of these, as well as their limits in terms of data fitting and constraint holding.

It is important to notice that the addition of the constraint term in (**??**) further reduces the effective degree of freedom of the model, similar as for P-splines, resulting in a less flexible model. We therefore expect that the measured metric on the training data will be worse compared to a pure B-spline fit. Nevertheless, the metric of interest is the measured error on the validation data, i.e. the held out data that the model has not seen before. Supposing that the a priori domain knowledge reflects the true, underlying function, we expect the measured error on the validation data to be lower than or equal to the error given by an optimal P-spline fit. Here, optimality is based on the optimal smoothing parameter λ given by generalized cross-validation, see Section **??**. This can be seen by recognizing the equivalence of the objective functions for P-splines, see (**??**), and shape-constraint P-splines, see (**??**), when the underlying B-spline fit does not violate the user-defined constraint, i.e. all $v_j = 0$. This feature is one of the limits of shape-constraint P-splines, i.e. we cannot influence the model using this approach if no constraint violations are present. On the other hand, if the a priori domain knowledge reflects the underlying function, we can expect a far better generalization capability of the model compared to the B-spline fits, especially in situations of noisy or sparse data.

In this chapter, we are going to discuss the practical incorporation of a priori domain knowledge based on the example of peak behavior, see Section 1.1. We will evaluate the performance of shape-constraint P-splines using sparse and noisy data and compare it to B-splines and P-splines. Further, we will discuss the effect of the constraint parameter λ_c, see Section 1.3.

2.1 Peak Constraint in Practice

We will now use shape-constraint P-splines and the a priori domain knowledge of peak behavior to fit the data $D = \{(x^{(i)}, y^{(i)}),\ i = 1, 2, \dots, n\}$. The data is artificially generated by random sampling of $n = 200$ points $x^{(i)}$ of the function f, i.e.

$$y^{(i)} = f(x^{(i)}) + \epsilon^{(i)} = \exp\big(-\frac{(x^{(i)} - 0.35)^2}{0.1}\big) + \epsilon^{(i)} \quad \text{for } x^{(i)} \in [0.1, 0.8], \tag{2.1}$$

with $\epsilon^{(i)}$ being Gaussian noise with mean $\mu = 0$ and variance $\sigma^2 = 0.01$. We randomly split the data in a training set D_t with 150 samples and a validation set D_v with 50 samples. At first, we use $d = 45$ B-spline basis functions of order $l = 3$ to fit a B-spline to the training data D_t. Then, we fit a P-spline using the same number of basis functions d and order l with an optimal smoothness parameter $\lambda_{opt} = 7.74$ chosen by generalized cross-validation. Finally, we enforce the peak behavior by using a shape-constraint P-spline using the same number of basis functions d and order l as well as the optimal smoothness parameter $\lambda_{opt} = 3.68$ and the constraint parameter $\lambda_c = 6000$ reflecting high trust in the a priori domain knowledge. The various fits are evaluated on the validation data D_v and shown in Figure 1.1. The mean squared errors on the validation data, as well as on the true, underlying function in (1.1), are given Table 1.1.

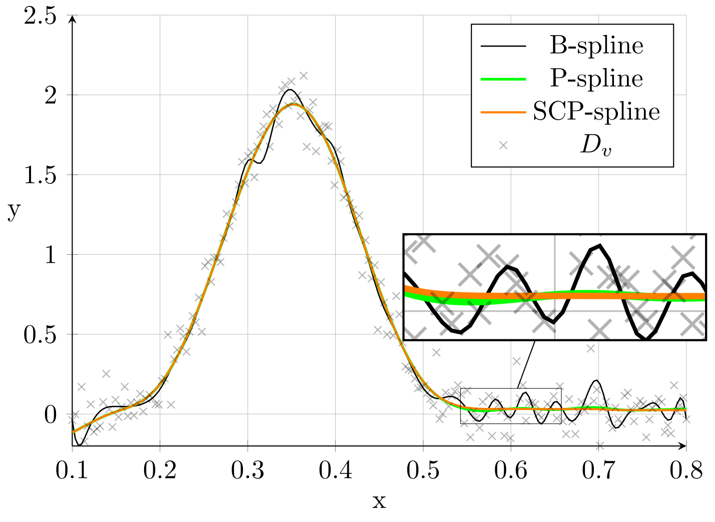

Figure 2.1: B-spline, P-spline and SCP-spline fit for D_v.

The B-spline, as black curve in Figure 1.1, captures the basic shape of the true function, but the flexibility of the B-spline, due to the high number of B-splines, leads to a wiggly curve especially for the almost constant part. This violates the peak behavior of the true function, i.e. being non-increasing after the peak value. For the P-spline, as green curve, this problem relaxes due to the smoothing aspect of the penalty term but does not vanish, as seen in the magnified part in Figure **??**. Note that the additional smoothness penalty given by the P-spline already fits the data almost perfectly. The SCP-spline fit adjusts only the parts of the P-spline, which violate the constraint. It may be seen as "fine-tuning" of the fit using the a priori domain knowledge. Hence, it is the best solution here as it is nearly constant for the necessary parts of the function in (1.1).

Model	MSE_{D_v}	$\text{MSE}_{D_{v,true}}$
B-spline	$1.9 \cdot 10^{-2}$	$7.04 \cdot 10^{-3}$
P-spline	$1.22 \cdot 10^{-2}$	$1.52 \cdot 10^{-3}$
SCP-spline	$1.22 \cdot 10^{-2}$	$1.48 \cdot 10^{-3}$

Table 2.1: Mean squared errors on the validation set.

The mean square errors on the noisy validation data D_v in Table 1.1 for P-spline and SCP-spline are almost identical and do not show a favorable model. Comparing the various models with the true, underlying function, see $\text{MSE}_{D_{v,true}}$ in Table 1.1, leads to the assessment that the SCP-spline is the more accurate model with regards to the true function behavior. This coincides with the graphs in Figure 1.1. Hence, the incorporation of a priori domain knowledge via shape-constraints improves the generalization capability measured by the mean squared error on the true function values.

2.2 Sparse Data and the Peak Constraint

We will now examine the behavior of B-, P- and SCP-splines for sparse data, i.e. little data and uneven distributed, sampled from the function in (1.1). The data set D now contains 70 data points distributed in a way that there is little data in the peak region, i.e for $x \in [0.2, 0.5]$ we have only 10 data points. We perform an random train-validation split of the data in D, i.e. the training data D_t consists of 52 points and the validation data D_v consists of 18 points. We follow the same approach as above, i.e. fit a B-spline, perform generalized cross-validation to fit the optimal P-spline and finally apply the shape-constraint to enforce peak behavior. The small data set indicates that a different, not equidistant knot placement may be helpful. Hence, we carry out 2 experiments, one with equidistant knot placement and the other with quantile-based knot placement, see Section **??**. We chose to use $d = 25$ B-spline basis functions of order $l = 3$. The optimal smoothness parameter was given as $\lambda_{opt} = 0.00657$ for the equidistant fit and $\lambda_{opt} = 0.00215$ for the quantile-based fit. The constraint parameter λ_c was set to $\lambda_c = 1000$ for both fits, reflecting high trust in the a priori domain knowledge. The resulting fits are shown in Figure 1.2 and Figure 1.3.

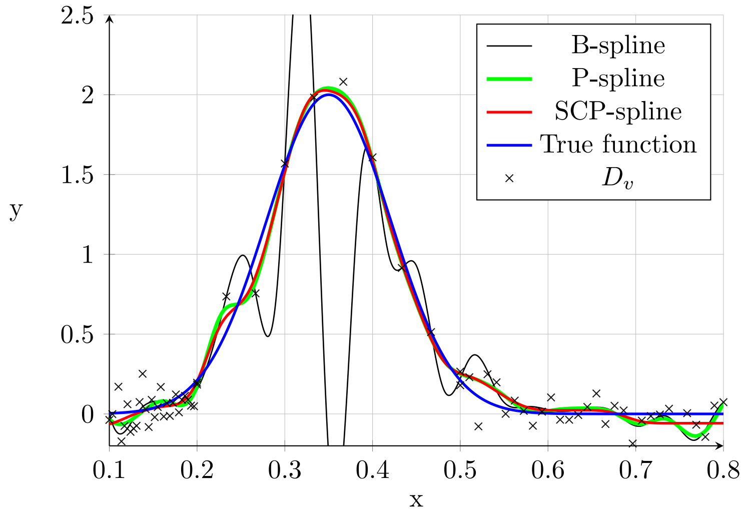

Figure 2.2: Equidistant B-spline, P-spline and SCP-spline fit for sparse data D.

The equidistant B-spline in Figure 1.2 shows the problem of B-splines in sparse data situations. The estimate (black) becomes very wiggly, similar to polynomial fits using a high degree. Nevertheless, using the regularization through the additional smoothness penalty in P-splines, the estimate (green) becomes quite smooth and reflects the true function quite well. The shape-constrataint P-spline (red) further improves the quality of the fit, as seen in Table 1.2.

Model	MSE_{D_v}	$\text{MSE}_{D_{v,true}}$
B-spline	0.32	0.27
P-spline	$1.44 \cdot 10^{-2}$	$3.94 \cdot 10^{-3}$
SCP-spline	$1.36 \cdot 10^{-2}$	$2.32 \cdot 10^{-3}$

Table 2.2: Mean squared errors on the validation set for equidistant knot placement.

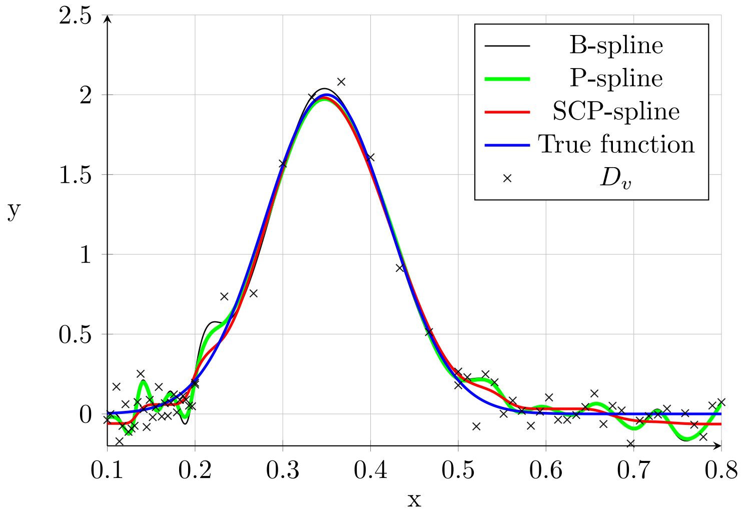

Figure 2.3: Quantile-based B-spline, P-spline and SCP-spline fit for sparse data D.

The quantile-based B-spline (green) in Figure 1.3 shows some interesting features. At first, it fits the peak surprisingly well despite the small sample size in this area. This is due to the inherent smoothing aspect of quantile-based knot placement in regions of small sample size. Further, we see the flexibility of the B-spline in the regions with more data, i.e. $x \in [0.1, 0.2]$ and $x \in [0.5, 0.8]$, where it clearly fits the noisy part instead of the true function. Hence, quantile-based knot placement may lead to partial overfitting. (further descriptions are needed here) Using a P-spline (green) does not relax this problem. This may be caused by the

Model	MSE_{D_v}	$\text{MSE}_{D_{v,true}}$
B-spline	$2.51 \cdot 10^{-2}$	$1.18 \cdot 10^{-2}$
P-spline	$2.31 \cdot 10^{-2}$	$9.32 \cdot 10^{-3}$
SCP-spline	$1.48 \cdot 10^{-2}$	$2.52 \cdot 10^{-3}$

Table 2.3: Mean squared errors on the validation set for quantile-based knot placement.

2.3 Lambda c

2.4 Something

In this chapter we use the theory discussed in Chapter Chapter **??** to estimate uni- and bivariate functions using data and a priori domain knowledge. An overview of the different problems considered in this chapter is given in Table Table 1.4.

Univariate	**Section**	**Bivariate**	**Section**
B-splines		Tensor-product B-splines	
P-splines		Tensor-product P-splines	
SCP-splines		Tensor-product SCP-splines	

Table 2.4: Problem overview.

First, we are using B-splines, see Section **??**, for the estimation of the unknown function $y = f(x)$, i.e. we solve the optimization problem

$$\arg\min_{\boldsymbol{\beta}} Q_1(\mathbf{y}, \boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|, \tag{2.2}$$

using the B-spline or tensor-product B-spline basis matrix $\mathbf{X}$. Next, we use the concept of P-splines, see Section **??**, to estimate smooth functions, i.e. we solve the optimization problem

$$\arg\min_{\boldsymbol{\beta}} Q_2(\mathbf{y}, \boldsymbol{\beta}; \lambda) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\| + \lambda \cdot \mathrm{pen}(\boldsymbol{\beta}), \tag{2.3}$$

where $\mathrm{pen}(\boldsymbol{\beta})$ specifies a smoothness penalty term. Finally, we are going to incorporate a priori domain knowledge into the fitting process using shape-constrained P-splines (SCP-splines), i.e. we solve the optimization problem

$$\arg\min_{\boldsymbol{\beta}} Q_3(\mathbf{y}, \boldsymbol{\beta}; \lambda, \lambda_c) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\| + \lambda \cdot \mathrm{pen}(\boldsymbol{\beta}) + \lambda_c \cdot \mathrm{con}(\boldsymbol{\beta}), \tag{2.4}$$

where $\mathrm{pen}(\boldsymbol{\beta})$ is again a smoothness penalty term and $\mathrm{con}(\boldsymbol{\beta})$ specifies the user-defined shape-constraint to incorporate a priori domain knowledge with, see [**hofner2011monotonicity**] and [**bollaerts2006simple**]. Various types a priori domain knowledge can be incorporated using the constraints listed in Table **??**.

The focus of this chapter is the definition and use of shape-constraint P-splines, which are characterize by their parameters $\boldsymbol{\beta}$ given by solving the optimization problem (1.4).

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, XX. Monat, Jahr

Vorname Nachname