Universität Bremen
Fachbereich 3 (Mathematik & Informatik)
Zentrum für Technomathematik

Tikhonov Functionals Incorporating Tolerances in Discrepancy Term for Inverse Problems

Dissertation

zum Erlangen des Grades eines Doktors der Naturwissenschaften

-Dr. rer. nat.-

vorgelegt von

Phil Gralla

1. Gutachter: Dr. Iwona Piotrowska-Kurczewski, Universität Bremen
2. Gutachter: Prof. Dr. Peter Maaß, Universität Bremen
3. Gutachter: Dr. Oltmann Riemer, Leibnitz-Institut für Werkstofforientierte Technologie **IWT** Bremen

Kolloquium: 13. Januar 2023

Abstract

This thesis contributes to the field of inverse problems and their regularization through Tikhonov-type regularization schemes. Tikhonov-type regularization schemes use a discrepancy term on the operator evaluation and a regularization term on the parameter. One can change the penalty term to incorporate different a-priori information about the true parameter. For example, most research focuses on the regularization term, such as applying sparsity constraints instead of L_2-penalty. This work takes a different approach and adds an ε-insensity to the discrepancy term. This insensitivity does add another regularization and accounts for confidence bands. We can obtain these bands through multiple measurements, for example.

Besides the mathematical framework, this work explores possible numerical solvers for the altered Tikhonov functional. Depending on the chosen norm of the discrepancy term and the type of penalty term, the altered Tikhonov functional may not be differentiable. In this case, a non-smooth solver is necessary. This thesis compares existing non-smooth solvers with a newly introduced subgradient method with adaptive step size.

Finally, we apply the theory to a parameter identification problem. The example is from micro-milling and the resulting surface. First, an existing cutting process model is taken and expanded to account for wear on the cutting tool during the process. Then we use the model for the parameter identification.

Acknowledgement

The author acknowledges the financial support by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for the CRC 747 (Collaborative Research Center) *Mikrokaltumformen – Prozesse, Charakterisierung, Optimierung.*

This project was supported by the Deutsche Forschungsgemeinschaft (DFG) within the framework of GRK 2224/1 *Pi^3: Parameter Identification - Analysis, Algorithms, Applications.*

I would like to thank

My supervisors Prof. Dr. Peter Maaß and Iwona Piotrowska-Kurczewski for advising, supporting and encouraging me.
My wife and daughter for their support and understanding.
My family for supporting me and believing in my ability to finish.
Christine, Delf, Georgia, Simon, Thilo, and Ziwen for the discussions and helpful inputs.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Abbreviation

CRC 747	Collaborative Research Center 747
SVM	Support Vector Machines
SVR	Support Vector Regression
w.c.	weakly continuous (operator)
w.l.s.c.	weakly lower semicontinuous (function)
w.s.c.	weakly sequentially closed (set or operator)
ZeTeM	Center for Industrial Mathematics at University of Bremen

Symbols

∇	gradient
$\mathsf{L}(X, Y)$	space of linear operators between X and Y
$\lvert \cdot \rvert$	absolute value
$\lVert \cdot \rVert_X$	norm in X
$\lVert \cdot \rVert_{X,\varepsilon}$	ε-insensitive premetric
$\mathcal{R}(A)$	image space of operator A
$\mathcal{D}(A)$	definition space of operator A
$\mathbb{N}$	space of positive natural numbers
$\mathbb{R}$	space of real numbers
$\mathbb{R}^+$	space of positive real numbers (excluding 0)
$\mathbb{R}_0^+$	space of positive real numbers including 0
$L^2(\Omega)$	space of square integrable functions on Ω
A	Fréchet-Differential of the operator F
$\lfloor \cdot \rfloor$	round down to next integer
$\lceil \cdot \rceil$	round up to next integer
$B_r(x)$	open ball with center at x and radius r
$d(\cdot, \cdot)$	metric
$\langle \cdot, \cdot \rangle$	inner product
$\odot$	Hadamard product
$\oslash$	Hadamard division

1 Introduction

An inverse problem seeks the solution of

$$Fu = v \tag{1.1}$$

where $F : X \to Y$ and $v \in Y$ are known and $u \in X$ is unknown. In many applications, either the true right side of the equation is unknown and instead a noisy version v^δ with $\|v - v^\delta\| \leq \delta$ or the operator F does not have an inverse or is ill-conditioned. This means that a solution is either not unique or the reconstruction $\hat{u}$ with

$$F\hat{u} = v^\delta$$

is not close enough to the true u, i.e. $\|u - \hat{u}\| \gg \delta$. In such a case we speak of an ill-posed inverse problem. A combination of the above may also be present. For an inverse problem it is important to find the true solution u of (1.1) and it is not enough to find a minimizer of

$$\operatorname*{argmin}_{u \in X} \left\| Fu - v^\delta \right\|.$$

To solve such problems different techniques of regularization exists that reduce the influence of noise in v and find an inverse or pseudo-inverse of F. One popular approach is Tikhonov regularization where

$$u^\dagger = \operatorname*{argmin}_{u \in X} \left\{ \left\| Fu - v^\delta \right\|_p^p + \alpha \mathcal{R}_q(u) \right\}$$

is an approximation of u under the regularization $\mathcal{R}_q(u)$. The works of Grasmair [28], Kaltenbacher [39] study the different ways of regularization with various regularizers $\mathcal{R}(u)$. While Kazimierski et al. [40], as well as Bangti and Maass [36] focus on sparsity constraints on the reconstructed u by restricting it to be a linear combination of wavelets, and the regularization is then applied to the coefficients of u. These are often sparsity constraints with $q = 1$.

In this work the Tikhonov regularization for inverse problems is altered by adding a tolerance insensitivity to the discrepancy term $\left\| Fu - v^\delta \right\|_p^p$. The

level of tolerance is defined by a parameter ε. As a result the problem at hand in this work is to calculate

$$u^\dagger = \operatorname{argmin} J^{p,q}_{\delta,\alpha,\varepsilon}(u)$$

where

$$J^{p,q}_{\delta,\alpha,\varepsilon}(u) = \frac{1}{p}\left\|F(u) - v^\delta\right\|^p_{L_p,\varepsilon} + \alpha\mathcal{R}_q(u). \tag{1.2}$$

We use a hinge loss for the ε-insensity, which is also used for training support vector machines for classification or regression.

The hinge loss is defined as

$$|\lambda|_\varepsilon := \begin{cases} 0 & |\lambda| \leq \varepsilon \\ |\lambda| - \varepsilon & \text{otherwise} \end{cases} \tag{1.3}$$

and can be extended for functions in $L_p(\Omega)$ via point wise evaluation. The result is a so-called $(\varepsilon\,, L_p)$-insensitive distance function defined as

$$\|f\|^p_{L_p,\varepsilon} = \int_\Omega |f(x)|^p_\varepsilon \, \mathrm{d}x \tag{1.4}$$

for all $f \in L_p$.

This change in the discrepancy will yield non-standard discrepancy terms which do not fulfill the standard requirements. As such, we prove the existence of a minimizer, stability for ε and δ and convergence before having a deeper look into methods to solve the minimization problem numerically. The advantage of changing the discrepancy term is to better model uncertainties in measurements and to further stabilize the reconstruction of the true solution $\hat{u}$.

Figure 1.1 depicts the relation of p, q and ε. Each point in the box stands for a triple of p, q, ε. The base plane is the classical theory for inverse problems without any tolerance for the discrepancy. The vertices on the top are known triplets in support vector machines. We focus on completing the cube in this work, and special attention is given to the transitions between triplets. Especially, the transitions along the z axis are important to unify the standard theory with the additional regularization in the discrepancy.

1.1 Support Vector Machines

Support vector machines are a method in machine learning for regression and classification. Machine learning is a mathematical field where models are derived from training data and applied to unknown data. These methods can be applied to many different problems as long as enough data is available. As a result, they offer a versatile tool to solve various problems.

The support vector method is a linear method that uses a hyperplane as a decision boundary for classification or prediction function for regression.

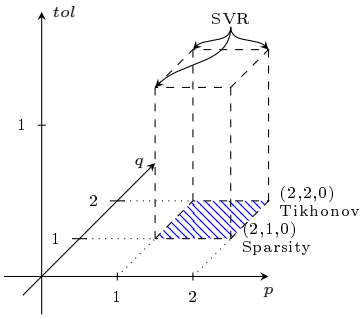

Figure 1.1: Diagram for Tikhonov Methods. Each point in the diagram represents a set of p, q combinations for Tikhonov type functionals. In addition $tol = 1$ denotes that there is a tolerance in the discrepancy term of the functional.

For complex models, it is either necessary to find a large feature space, i.e., transform the data into a new space on which the linear support vector method is trained, or use kernels to introduce non-linearity to the linear model. These kernels need to be reproducing kernels in Hilbert spaces, and their usage in support vector machines is known as *kernel trick*. This technique is not unique to support vectors and can be applied to many regression techniques. The properties of reproducing kernels in Hilbert spaces allow to directly compute the solution without the need to calculate and transform the input vectors explicitly.

The name, support vector machines, is derived from the fact that a subset of the training data defines the separating hyperplane. This subset is the support vectors. For a two class classification, the support vector decision boundary is defined by

$$L_{w,\beta_0} := \{x \in X \mid x \in \langle w, x\rangle + \beta_0 = 0\},\ w \in X, \beta_0 \in \mathbb{R}$$

and the decision function is

$$f(x) = \operatorname{sgn}\left(\langle w, x\rangle + \beta_0\right)$$

where the classes are represent with $\{-1, 1\}$ respectively. An example for classification is shown in figure 1.2. Two core elements of the support vector method are the loss function and regularization. The loss function is the hinge loss defined as

$$|x|_\varepsilon := \max\{0,\ |x| - \varepsilon\}$$

for $x \in \mathbb{R}$. For finding a unique hyperplane for classification or regression a regularization is needed. For regression with support vector machines the regressor $f(x) = \langle w, x_i\rangle + \beta_0$ is the solution to the minimization problem

$$\underset{w,\beta_0}{\operatorname{argmin}} \sum_{i=1}^{n} |\langle w, x_i\rangle + \beta_0 - y_i|_\varepsilon^2 + \alpha\left(\|w\|^2 + |\beta_0|\right)$$

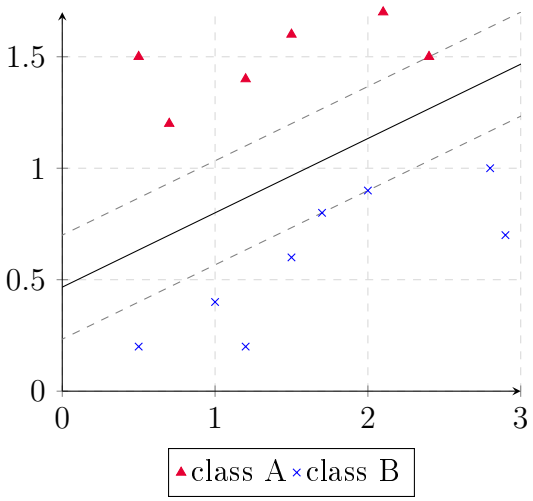

Figure 1.2: A toy example for SVM classification. The two classes are marked with red triangles and blue stars. The solid line marks the decision boundary $\langle w, x\rangle + \beta_0$ and the dotted lines the margin.

over the training data $\{(x_i, y_i)\}$. Rewriting the above minimization with $f(x) = \langle w, x_i\rangle + \beta_0$ yields

$$\operatorname*{argmin}_f \sum_{i=1}^{n} |f(x_i) - y_i|_\varepsilon^2 + \alpha \|f\|^2$$

and demonstrates the connection between Tikhonov functionals and support vector regression.

1.2 Machine Learning for Inverse Problems

In recent years, machine learning techniques, especially deep learning types, have been studied and used to solve applications and tasks that are closely related or are part of inverse problems.

In inverse problems, we need to find a parameter u such that we minimize

$$\|Fu - v\|$$

and that, at the same time, we minimize

$$\|u - \hat{u}\|$$

where

$$F\hat{u} = v$$

and $\hat{u}$ is the ground truth. As stated before, the second condition is crucial for inverse problems and separates them from optimization problems.

For machine learning, we want to find a function/operator $F : X \to Y$ such that we minimize

$$\|Fu_i - v_i\| .$$

Machine learning methods can directly solve an inverse problem if we have data pairs u_i, v_i with a known operator $Fu_i = v_i$. Then the operator

$$\hat{G} = \operatorname*{argmin}_G \sum_i \|Gv_i - u_i\|$$

approximates the pseudo inverse of F and can be used to solve the inverse problem for a new data v. If enough training data is available, G is a suitable approximation for the pseudo-inverse of F. This technique is used with variational autoencoders in [44, 61], for example. Another application is image generation. Ho et al. train diffusion models to generate an image from noise and solve an inverse problem in the process [32]. Saharia et al. use the same diffusion base networks for image super-resolution [64].

The downside of this approach is the necessary number of samples needed for finding a stable inversion of F. We can generate or enhance the training sets $\{u_i, v_i\}$ with artificial data to ensure the stability of $\hat{G}$ on new unseen data if enough information about the expected noise and forward operator is known.

A second approach using machine learning to solve inverse problems is restricting the solution space X. For example, Ulyanov et al. [80] and Sitzman et al. [71] use deep image priors to solve inverse problems in image reconstruction to restrict the space of possible solutions to images generated by U-networks, a particular architecture of deep neural networks. This approach is similar to Morozov regularization [57].

This work does take elements from the support vector machines and integrates them into the existing Tikhonov-type methods. Therefore, it is an indirect approach to combine inverse problems with techniques from machine learning while avoiding the need for training data or to find a suitable network architecture to restrict the solution space.

1.3 Numerical Methods

We extend the classical theory of inverse problems in this paper by proving mathematical prerequisites such that a solution exists and is obtainable. We can use existing numerical methods to find the desired minimizer numerically for some cases, such as $p = 2$ and a convex regularization term. To complete the theory and allow us to use our regularization in real applications, we extend existing numerical solvers to solve for a more significant number of regularization terms and other values for p. We are especially interested in the case where $p = q = 1$.

The minimization function is not differentiable everywhere for $p = q = 1$, and we can not use standard convex solvers to find the desired minimum. If we restrict the forward operator F to linear and the regularization term $\mathcal{R}_q(u) = \|u\|_q^q$, we can apply solvers used for fitting support vector machines to solve the inverse problem. We need a different numerical solver for more regularization terms or non-linear forward operators.

All numerical methods minimize a function $f(x)$. Stochastic gradient descent methods use a statistical approximation of a true (sub-)gradient to minimize the function iteratively. They are widely used to train neural networks and other machine learning algorithms when working with batches or mini-batches. (Mini-)batches mean that the dataset is not processed in one batch to calculate the (sub-)gradient but rather a subset. This batching gives a statistical approximation of the (sub-)gradient. In addition, many modern methods use momentum and adaptive step size in their update rules. It is important to note that while stochastic gradient methods for subgradients exist, not all stochastic methods are subgradient methods. ADAM, for example, is only proven to converge for gradients [43].

Subgradient methods generalize the gradient to be defined on a convex function that is non-differentiable via

$$f(y) \geq f(x) + g^T(x - y)$$

where g is not unique. As such, subgradient methods can be used to find a minimizer for the non-differentiable Tikhonov functional. They do, however, rely on an a-priori step-size rule and need many iterations to converge. We introduce an adaptive step-size rule for the classical subgradient method to reduce the downside of the subgradient methods.

Gradient-free methods do not need any (sub-)gradient information and do not need convexity of the minimization function. They can minimize a function if it has many local minima and is non-smooth. The Nelder-Mead simplex algorithm and Particle Swarm optimization are two examples of gradient-free methods. These methods need many iterations to converge and are limited to small input spaces to be feasible.

1.4 Micro Cold Forming

We originally designed the presented mathematical methods in this work to be applied to parameter identification for micro-milling as part of the Cooperative Research Center 747 (CRC 747) at the University of Bremen. In the CRC 747, new methods for cold forming of micro parts in micro dimensions are studied. Micro dimensions means that the parts are smaller than 2mm in two dimensions. Figure 1.3 shows one example of a micro-scale component next to a grain of rice for size comparison.

Insignificant effects on the macro-scale have a strong influence on the micro-scale. We call these size effects and Vollertsen further elaborated on size effects in [86]. Size effects hamstring the transition of the manufacturing process from macro-scale to micro-scale. For example, in milling processes, small structures on the surface of the finished workpiece are inevitable. These structures are crucial for microstructured devices or functional tribological surfaces [58, 6, 23].

We take the micro-milling process model from [60], and [83] to simulate a micro-milling cutting process. During the process, stress and pressure on the

Figure 1.3: Micro cup next to a grain of rice for size comparison. (Picture provided by CRC 747, photographed by Thorsten Müller)

cutting edge cause wear. This wear changes the shape of the cutting edge and thus the microstructures of the finished surface. We add a general model for the change in geometry caused by wear to the existing process model.

The Abbott-Firestone curve, also called area bearing curve, defines surface characteristics in a one-dimensional graph. This graph captures the local and global surface structures. We use this graph to define the desired surface based on necessary properties. In addition, we can directly define a tolerance on this graph for the surface to still have all desired properties.

The introduced tolerances allow defining surfaces with acceptable tolerance bands, which we use to find ideal process parameters considering wear on the cutting tool over time.

1.5 Overview about the Thesis

The thesis is structured as follows. First, the preliminary chapter gives necessary basic definitions and mathematical tools. Then, we focus on essential topics for proofs and concepts used in this thesis. Necessary fundamentals include convex analysis, functional analysis, continuities, statistical distributions, noise models, ill- and well-posed problems, and convex optimization. For readers familiar with any of the mentioned topics, the chapter should only establish notations but not introduce new concepts. Later chapters will provide more advanced definitions and theorems in this work when needed or applied.

Chapter 3 outlines tolerances and support vector machines in Hilbert spaces. We introduce the classical support vector machines (SVM) and support vector regression (SVR) and their extension with kernels in reproducing kernel Hilbert spaces. In addition, we explore their relation to Tikhonov regularization. Furthermore, we will establish a general mathematical framework to utilize regularization via tolerance in discrepancy terms for Tikhonov regularization. The necessary proofs for existence, stability, and convergence are

derived and explained in detail. For later applications, the L_p spaces need to be included. An example of these applications is part of chapters 3 and 4.

Chapter 4 addresses numerical solutions and implementation of the altered Tikhonov functional. We explore existing numerical methods before developing a novel subgradient method with adaptive step-size. This subgradient method is similar to the standard methods, but increases converge speed through online step-size adaptation. The algorithm is then applied and compared to other regularization methods in two signal and image processing examples.

The final chapter presents the application for micro-milling. First, we give the necessary background and modeling of micro-milling processes and explain the application task. Afterward, we use our method to solve the task at hand.

1.6 Code and Supplementary

The code, images and examples from this thesis are public and available on the github repository
`https://github.com/PGralla/PhD-Thesis-Supplementary`.

2 Preliminaries

This chapter provides an overview of mathematical concepts and notations used in this thesis. First, we give basic notations and definitions used in this thesis. Then, in the following two sections, concepts of convex analysis, and different types of continuity, as well as closeness in a weak sense, are introduced. Additionally, in section 2.6 a short overview of data analysis and statistics is given. These concepts will help us understand the origin of tolerances and where we derive the idea for altering Tikhonov functionals from in chapter 3. Finally, the last two sections cover inverse problems and their regularization, primarily focused on Tikhonov Methods and numerical optimization, which is needed to solve minimization problems numerically.

2.1 Basic Notations and Definitions

The symbol $\mathbb{K}$ is a synonym for the scalar field of real numbers $\mathbb{R}$ or complex number $\mathbb{C}$. Furthermore, we will denote by $|\Omega|$ the n-dimensional Lebesque measure of $\Omega \subset \mathbb{R}^n$. For any measurable subset $\Omega \subset \mathbb{R}^n$ the *Lebesque space* L_p with $0 < p \leq \infty$ consists of all measurable function f, i.e.

$$\|f \mid L_p\| = \|f\|_{L_p} = \left(\int_\Omega |f(x)|^p \, \mathrm{d}x \right)^{1/p} \tag{2.1}$$

is finite. In the limiting case $p = \infty$, the usual modification with the essential supremum norm is required. We consider $\mathbb{K}$ valued function $f \in L_p$. Operators $F : X \to Y$ between vector spaces are referred to as *functionals* if $Y = \mathbb{R}$. If $A : X \to Y$ is a linear operator between vector spaces, we refer to it as a *linear functional* if $Y = \mathbb{R}$.

The operator equation is

$$F : U \to L_p(\Omega) \tag{2.2}$$

for U Hilbert-space and $1 \leq p \leq 2$.

The Tikhonov-functional

$$J_{p,q}^{\delta,\alpha}(u) = \frac{1}{p}\left\|F(u) - v^{\delta}\right\|_{L_p}^{p} + \alpha\mathcal{R}_q(u) \tag{2.3}$$

is under consideration.

2.2 Convex Analysis

This section covers the basics of convex analysis used in this thesis. Further information can be found in [68]. However, we first must define what convex sets and functions are.

DEFINITION 1 (*Convex Set*)

Let X be a vector space and $M \subset X$. The set M is convex if

$$\lambda x + (1-\lambda)y \in M,\ \forall x, y \in M,\ \lambda \in (0,1)$$

A particular subset of convex sets are radially convex sets. We use these sets to model the micro-milling process and the wear on the cutting tool.

DEFINITION 2 (*Radially Convex Set*)

Let X be a vector space and $M \subset X$. The set M is radially convex if there exists at least one $x_0 \in M$ s.t.

$$\lambda x_0 + (1-\lambda)y \in M,\ \forall y \in M,\ \text{and } \lambda \in (0,1)$$

holds.

Radially convex sets are also called star-shaped sets or star-convex sets. All convex sets are radially convex, which follows directly from the definitions of convex and radially convex sets. For functionals, convexity is defined as follows.

DEFINITION 3 (*Convex Functional*)

Let M be a convex subset of a vector space, $x, y \in M$, and $\lambda \in [0,1]$. The functional $f : M \to \mathbb{R}$ is called

***convex** if*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

***strictly convex** if*

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y).$$

We use epigraphs of functionals to analyze functionals and in several proofs later in this thesis.

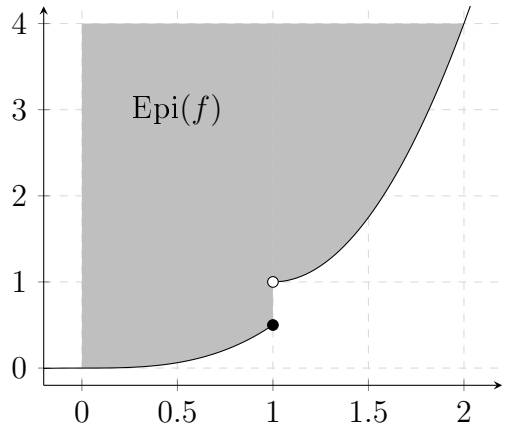

Figure 2.1: Example of an epigraph for a piecewise differentiable function on the domain $x \in [0, 2]$. The epigraph is the grey area including the graph of the function itself.

DEFINITION 4 (*Epigraph*)

Let X be a vector space and $f : X \to \mathbb{R}$ a functional. The epigraph of the functional f is defined as

$$Epi(f) := \{(y, x) \in \mathbb{R} \times X \mid y \geq f(x)\}$$

and is the set of all points above the graph including itself.

Figure 2.1 shows an example of an epigraph. Convex sets and convex functionals are related directly through the epigraph. We use the following proposition for this relation.

PROPOSITION 5

A functional f is convex if and only if the epigraph of f is convex.

Proof: First we assume that a functional $f : X \to \mathbb{R}$ is convex. Let (y_1, x_1) and (y_2, x_2) be in the epigraph of f. Define the point

$$(\hat{y}, \hat{x}) := \lambda(y_1, x_1) + (1 - \lambda)(y_2, x_2)$$

with $\lambda \in [0, 1]$.

Then by the definition of epigraph and convexity

$$\begin{aligned} \hat{y} &= \lambda y_1 + (1 - \lambda) y_2 \\ &\geq \lambda f(x_1) + (1 - \lambda)(x_2) \\ &\geq f(\lambda x_1 + (1 - \lambda)(x_2)) \\ &= f(\hat{x}) \end{aligned}$$

and thus the epigraph is convex.

Now let us assume that the epigraph is convex. Let $x_1, x_2 \in X$ be arbitrary. Then

$$(f(x_1), x_1), (f(x_2), x_2) \in \text{Epi}(f).$$

Since the epigraph is convex the point

$$\lambda (f(x_1), x_1) + (1 - \lambda) (f(x_2), x_2) \in \text{Epi}(f)$$

for all $\lambda \in [0, 1]$. By the definition of the epigraph

$$\lambda f(x_1) + (1 - \lambda) f(x_2) \geq \lambda x_1 + (1 - \lambda) x_2$$

and thus f is convex, which completes the proof. ■

The epigraph of a functional also has unique properties for convex and lower semicontinuous functionals.

PROPOSITION 6

If f is convex and lower semicontinous, the epigraph is convex and strongly closed.

Proof: From proposition 5 we know that $\text{Epi}(f)$ must be convex. Let $y < f(x)$, then since f is lower semicontinous $(y, x) \notin \text{Epi}(f)$. ■

Convex functionals and convex sets are essential in convex optimization and variational analysis. One important theorem is the Hahn-Banach separation theorem. We use the following version of the Hahn-Banach separation theorem.

PROPOSITION 7 (*Hahn-Banach Separation Theorem*)

Let X be a topological vector space over K, where K is $\mathbb{R}$ or $\mathbb{C}$. If $A, B \subset X$ are both convex, non-empty and disjoint, then

- *if A is open, then there exists a continuous linear map $\varphi : X \to K$ and $t \in \mathbb{R}$ such that $Re(\varphi(a)) < t \leq Re(\varphi(b))$, $\forall a \in A$ and $b \in B$*
- *if V is locally convex, A is compact and B is closed, then there exists a continuous linear map $\varphi : X \to K$ and $s, t \in \mathbb{R}$ such that $Re(\varphi(a)) < t < s < Re(\varphi(b))$, $\forall a \in A$ and $b \in B$.*

In some literature, the theorem is stated in an alternative way, as above, by using separating hyperplanes instead of using linear mappings. In addition, there are more versions of the Hahn-Banach theorem for geometry, also known as Mazur's theorem, and other vector spaces. This thesis uses the Hahn-Banach separation theorem to refer to the above definition.

2.2.1 Derivatives and Subgradients

For the numerical minimization of functionals we need derivatives. Derivatives define (local) minima and provide an update direction for iterative numerical minimizers. The introduced types of derivatives are independent of the convexity of the considered functional. We consider three main types of derivatives. The directional, Gâteaux and Fréchet derivative. They are all defined as below.

DEFINITION 8 (*Directional derivative*)

Let X and Y be Banach Spaces and $f : X \to \mathbb{R} \cup \{-\infty, \infty\}$. The directional derivative of f at x in direction h is defined by

$$f'(x;h) = \lim_{t \searrow 0} \frac{f(x+th) - f(x)}{t}$$

if the limit exists.

The Gâteaux derivative is defined based on the directional derivative.

DEFINITION 9 (*Gâteaux derivative*)

Let X and Y be Banach Spaces and $f : X \to \mathbb{R} \cup \{-\infty, \infty\}$. If a $x^* \in X^*$ exists such that

$$f'(x;h) = \langle x^*, h \rangle \quad \forall h \in X,$$

then f is called Gateaux differentiable at x and $f'(x) = x^*$ is called the Gâteaux derivative of f at x.

DEFINITION 10 (*Fréchet derivative*)

If a $x^* \in X$ and $\varepsilon > 0$ exists such that an expansion of the form

$$f(x+h) = f(x) + \langle x^*, h \rangle + o(\|h\|)$$

holds for all h with $\|h\| < \varepsilon$, then f is called Frechet differentiable at x and $f'(x) = x^*$ is called the Fréchet derivative of f at x.

While the two definitions are similar not every Gâteaux differentiable is Fréchet differentiable. A generalization of derivatives for convex but non-smooth functions is subgradients.

DEFINITION 11 (*Subgradient and Subdifferential*)

Let M be a convex subset of a real Banach space X and $f : M \to \mathbb{R}$ a convex functional. An element $x^* \in X^*$ is a subgradient of f in x_0 if

$$f(x) \geq f(x_0) + \langle x^*, x - x_0 \rangle \quad \forall x \in M$$

and $f(x_0) \neq \pm\infty$. The subdifferential of f at x_0 is denoted as $\partial f(x_0)$ defined

as

$$\partial f(x_0) := \{x^* \in X^* \mid f(x) \geq f(x_0) + \langle x^*, x - x_0\rangle \quad \forall x \in M\}$$

which is the set of all subgradients of f in x_0.

For algebraic operators, subgradients are defined as stated below.

DEFINITION 12 (*Algebraic Subdifferential*)

Let $M \subset X$ be a convex subset of a real Banach space and $F : M \to Y$ be a convex Operator between Banach spaces M and Y. A linear Operator $A \in L(X, Y)$ is a algebraic subgradient at $x_0 \in M$ if

$$F(x_0 - h) - F(x_0) \geq A(h), \forall h \in X, \text{ such that } x_0 - h \in M$$

holds and $F(x_0) \neq \pm\infty$. The algebraic subdifferential of f at x_0 is denoted as $\partial_a F(x_0)$ and defined as

$$\partial_a F(x_0) := \{A \in L(X, Y) \mid F(x_0 - h) - F(x_0) \geq A(h),\ \forall h \in X, s.t. x_0 - h \in M\}.$$

Similar to gradients, subgradients provide a way to characterize local minima. The following lemma states the relation between subgradients and minima.

LEMMA 13 (*Minimum*)

Let $\mathcal{D}(f)$ be convex and a subspace of a real Banach space, f convex and $x \in \mathcal{D}(f)$. Then

$$0 \in \partial f(x) \quad \Leftrightarrow \quad f(x) \text{ is a minimum.} \tag{2.4}$$

Proof: First we show that from $0 \in \partial f(x)$ follows that $f(x)$ is a minimum. Let $0 \in \partial f(x)$ and $y \in X$ such that $x + y \in \mathcal{D}(f)$, then

$$f(x + y) \geq f(x) + \langle 0, y\rangle = f(x).$$

Since y is arbitrary and $\mathcal{D}(f)$ convex, each point in $\mathcal{D}(f)$ can be represented with $x + y$ and thus $f(x)$ is a minimum.

To show the reverse direction let $f(x)$ be a minimum. Therefore $\forall y \in \mathcal{D}(f)$:

$$f(x) \leq f(y).$$

Since $\mathcal{D}(f)$ is convex, there exists $z \in \mathcal{D}(f)$ such that $y = x + z$ and the inequality can be rewritten to

$$\begin{aligned} f(x + z) &\geq f(x) \\ &= f(x) + \langle 0, x + z\rangle \end{aligned}$$

which means the 0 element is a subgradient. ■

REMARK 14

We only assumed f to be convex and not strictly convex. Therefore the argmin might not be unique.

2.3 Different Types of Continuities

In this work, continuity in different spaces and settings is considered. Here an overview of the different considered continuities is provided.

DEFINITION 15 (*weakly lower semicontinuous*)

A function $f : M \subset X \to \mathbb{R}$ where M is a convex subset of the normed space $(X, \|\cdot\|)$ is weakly lower semicontinuous if for a weak convergent sequence $(u_k)_{k\in\mathbb{N}}$ to a point $u \in M$ holds

$$f(u) \leq \liminf_{u_k \to u} f(u_k).$$

DEFINITION 16 (*weakly continuous operator*)

An operator $F : U \to V$ is weakly continuous if for a weak convergent sequence $(u_k)_{k\in\mathbb{N}}$ to the point $u \in \mathcal{U}$, the sequence $(F(u_k))_{k\in\mathbb{N}}$ converges weakly towards $F(u)$.

Definitions that are needed later.

DEFINITION 17 (*weakly sequentially closed operator*)

An operator $F : U \to V$ is weakly sequentially closed if for any weakly convergent sequence $(F(u_k))_{k\in\mathbb{N}} \subset V$ the limit point lies in the range of F, i.e. there exists $u \in dom(F)$ such that $F(u_k) \rightharpoonup F(u)$.

DEFINITION 18 (*weakly sequentially closed set*)

A non-empty set $M \subset X$ of the normed space $(X, \|\cdot\|)$ is weakly sequentially closed if and only if for each weak convergent sequence $(x_n)_{n\in N} \subset M$ with limit x, x is an element of M.

PROPOSITION 19

Let X be a real Banach space and $f : X \to \mathbb{R} \cup \{-\infty, \infty\}$ be a convex functional. Then the functional f is lower semi-continuous if and only if it is weakly lower semi-continuous.

2.4 Ill-Posed Problems and Regularization

Inverse problems are based on functional analysis and involve finding a cause/-source for a known solution and operator. For finite spaces and linear models,

it is equivalent to solving an algebraic system of the form

$$Ax = b$$

for a known matrix $A \in \mathbb{R}^{n \times m}$ and solution vector $b \in \mathbb{R}^n$. However, the solution space may be infinite, and operators may not be linear. Typical fields of application are parameter identification, image processing, and image reconstruction. One classic example is computer tomography (CT), where methods of inverse problems are used to reconstruct the images of the inside of a patient that are non-invasive [63].

This section provides a small overview. For further information, we refer to [63] and [53]. The format of this section is adapted from [63]. The emphasis is on basic definitions and the *Tikhonov-Regularization*.

Let $F : U \to Y$ be an operator mapping from the parameter space U to the data space V. The known data $v^\delta \in V$ has a noise level $\delta \in \mathbb{R}$ which describes the variance of v^δ to the true data v. In this case the inverse problem can be formulated as a minimization problem

$$u_{sol} = \text{argmin} \|Fu - v^\delta\|$$

where in the absence of noise, u_{sol} should be the true solution u, i.e.

$$\|u_{sol} - u\| = 0.$$

An essential criterion is to estimate the distance between the computed solution and the real solution to characterize whether the operator F is ill-posed or well-posed. There is more than one definition for ill-posed problems. In this thesis, we use Hadamard's definition.

DEFINITION 20 (*Well posed problem (Hadamard)*)

Let U, V be Hilbert spaces, $F \in \mathsf{L}(U, V)$, $u \in U$ and $v \in V$. The Problem $F(u) = v$ is called well posed if

1. *$\forall v \in V$ exists a solution*
2. *each solution is unique*
3. *an inverse operator $F^{-1} : V \to U$ exists and is continuous.*

If the problem is not well posed it is said to be ill-posed.

The third requirement is sometimes expressed as *slight deviations in data result in minor deviations in u*. Most ill-posed inverse problems will violate this requirement, thus making them ill-posed.

For linear problems in Hilbert spaces we have a global characterization of ill-posed problems according to Nashed. For $F \in L(X, Y)$ we have the equivalent relation

$$\begin{aligned} Fu = v \text{ is ill-posed } &\Leftrightarrow \mathcal{R}(F) \text{ is not closed in } V \\ &\Leftrightarrow F^+ \text{ is not continuous.} \end{aligned}$$

If the linear problem has a compact operator, it is ill-posed. To solve an ill-posed problem, we minimize the discrepancy

$$\|Fu - v\| \tag{2.5}$$

instead. The argmin of (2.5) is not unique in general. To define a unique solution, we use the minimum norm element, which is called the generalized solution of

$$Fu = v.$$

Calculating the argmin of (2.5) is equivalent to solving the normal equation. First, we need to define adjoint operators.

DEFINITION 21 (*Adjoint Operator*)

Let $F \in L(U, V)$ be a linear operator between two Hilbert-spaces. The linear operator $F^* \in L(V, U)$ for which

$$\langle Fu, v\rangle_V = \langle u, F^*v\rangle_U$$

holds for all $u \in U$ and $v \in V$ is called the adjoint of F.

The adjoint operator is then used in the normal equation

$$F^*Fu = F^*v.$$

As stated before, solving the normal equation is equivalent to finding the argmin of (2.5). A linear operator that maps directly to the minimum norm element is called the Moore-Penrose-Inverse or generalized Inverse.

DEFINITION 22 (*Moore-Penrose-Inverse*)

The linear operator F^+ which maps each v to the minimum norm element of the set

$$\mathbb{L}_v = \{u \in \mathcal{D}(F) \mid F^*Fu = F^*v\}$$

is called the Moore-Penrose-Inverse or generalized inverse.

Instead of finding the minimum norm element, we can generalize using a function $\mathcal{R}_q : \mathcal{D}(F) \to \mathbb{R}^+$ and use this function to define the $\mathcal{R}_q$-minimizing solution.

DEFINITION 23 ($\mathcal{R}_q$*-minimizing solution*)

Let $\mathcal{R}_q$ be a functional that is defined on $\mathcal{D}(F)$ and maps into the $\mathbb{R}^+$. Then, $u^\dagger$ is called an $\mathcal{R}_q$-minimizing solution of $Fu = v$ if

$$u^\dagger = \operatorname*{argmin}_{Fu=v} \mathcal{R}_q(u).$$

well-posed problem	ill-posed problem
Multiplication by a small number $c \ll 1$	division by a small number
$c \cdot x = y$	$x = c^{-1}y$
integration	differentiation
$F(x) = F(0) + \int_0^x f(\tau)d\tau$	$f(x) = F'(x)$

Table 2.1: Examples for well- and ill-posed problems

An example for $\mathcal{R}_q$ is $\mathcal{R}_q(u) = \|u_0 - u\|^2$ where $u_0 \in U$ is a fixed element.

Table 2.1 lists a few examples for well- and ill-posed problems. These examples of well- and ill-posed problems are taken from [38]. As can be seen in these examples, a forward problem may be well-posed, but the inverse problem may not be, i.e., integration as a forward model is well-posed, but its inverse problem, differentiation, is ill-posed. Particular emphasis is on the continuous dependence of the problem since, in reality, not the exact solution g is known, but rather a noisy version denoted with g^ε. Possible causes are noise in the measurement methods (measurement error) and uncaptured physical phenomena in the mathematical model (model error). In such a case, we need the so-called regularization of the problem.

2.4.1 Tikhonov Type regularization

Tikhonov and Phillips introduced one of the first regularization methods. They introduced the Tikhonov functional

$$J^{p,q}_{\alpha,\delta}(u) = \left\|F(u) - v^\delta\right\|^p_Y + \alpha \left\|u - u^*\right\|^q_X, \ \alpha > 0 \tag{2.6}$$

which is being minimized over the domain of F. This functional takes known information u^* about the solution into account via a penalty term. Many variations of this penalty function exist and maybe more suitable for different problems.

The regularization of a Tikhonov functional is stated in [31] theorem 20.4. We state the theorem below for the case $p = q = 2$ and $u^* = 0$ of (2.6).

THEOREM 24

Let u^δ_α denote the unique minimizer of (2.6) with $p = q = 2$, $u^* = 0$, $v \in Im(F)$, and $\|u - u^\delta\| \leq \delta$. If $\alpha = \alpha(\delta)$ is chosen such that

$$\lim_{\delta\to 0} \alpha = 0 \quad \textit{and} \quad \lim_{\delta\to 0} \frac{\delta^2}{\alpha} = 0$$

then

$$\lim_{\delta\to 0} \|u^\delta_\alpha - u^\dagger\| = 0$$

Different choice of $\mathcal{R}_q$ for Tikhonov type regularizations exists. In this work we focus on minimum norm, sparsity, and total variation (TV) regularization. For norm regularization $\mathcal{R}_q(\cdot) = \|\cdot\|_q^q$. Let $\{\varphi_i\}_{i\in\mathbb{N}}$ define a basis of $\mathcal{D}(F)$, i.e. there exists $c_i \in \mathbb{R}$ for each φ_i such that

$$u = \sum_i c_i \varphi_i$$

for all $u \in \mathcal{D}(F)$. For sparsity constrains we want to regularize the solution on the c_i. For $\mathcal{R}_q$ we now choose

$$\mathcal{R}_0 = \sum \|c_i\|_0$$

which is the number of non-zero elements in $c = \{c_i\}_{i\in\mathbb{N}}$. For numerical stability and to be able to use numerical solvers instead of the index norm

$$\mathcal{R}_1 = \sum |c_i|$$

is choosen. This is an approximation of the sparsity norm but has better numerical properties.

The last regularization we consider in this work is the total variation. For this regularization

$$\mathcal{R}_q(u) = \|\nabla u\|_q^q$$

with $q \leq 1$. Other regularization terms exist, but we only focus on these three in this work.

2.4.2 Nonlinear Ill-Posed Problems

For non-linear operators, well- and ill-posedness needs to be defined locally. The following definition is based on Hadamard and is provided in [63].

DEFINITION 25 (*(Locally) Well- and ill-posed Problems*)

Let U and V be Hilbert spaces and $F : \mathcal{D}(F) \subset U \to V$ a be a (non-linear) operator. The operator equation $F(u) = v$ is said to be ill-posed in $u^\dagger \in \mathcal{D}(F)$ if for every $r >$ there exists a sequence $(u_k)_{k\in\mathbb{N}} \subset B_r(f^\dagger)$, such that

$$\lim_{k\to\infty} \left\|F(u_k) - F(u^\dagger)\right\|_V = 0, \text{ but } u_k \nrightarrow u^\dagger \text{ for } k \to \infty.$$

If an operator equation is not ill-posed it is said to be well posed.

We can apply this definition to linear operators as well. For example, a linear operator equation, $F(u) = v$ is well-posed if it is well-posed in every point, that is, if and only if F is injectiv and the image of F, $im(F)$, is closed in V.

2.5 Optimization

In optimization, a given function, f, must be minimized. We will only focus on minimization since the maximization of a function can be transformed into a minimization problem by multiplying it with a negative factor of one. For further information about optimization, we refer to [2, 37]. The basic case for optimization is the minimization of a convex function with convex conditions.

DEFINITION 26 (*Convex Minimization Problem*)

Let $f, g_i, h_j : \mathcal{D} \subset U \to \mathbb{R}$ with $i \in \{1, \dots, m\}$, $j \in \{1, \dots, p\}$ and U be a Hilbert space. The minimization problem

$$\begin{aligned} \underset{x \in \mathcal{D}}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \\ & h_j(x) = 0 \end{aligned} \tag{2.7}$$

is called convex if f, g_i are convex for all $i \in \{1, \dots, m\}$ and h_i is linear for all $j \in \{1, \dots, p\}$.

This definition is also referred to as standard form of a convex optimization problem or primal problem.

Solving the primal problem with constraints can be difficult. One way to find a solution is to solve an auxiliary problem instead. Under certain conditions, such an auxiliary problem has the same minimum but is simpler in its formulation and thus easier to solve. For example, one auxiliary problem is the Lagrange dual, which is defined below.

DEFINITION 27 (*Lagrange Dual Problem*)

Let $f, g_i, h_j : \mathcal{D} \subset U \to \mathbb{R}$ with $i \in \{1, \dots, m\}$, $j \in \{1, \dots, p\}$ and U be a Hilbert space. Then the minimization

$$\begin{aligned} \underset{\lambda, \mu}{\text{maximize}} \; \underset{x \in \mathcal{D}}{\text{minimize}} \quad & f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{2.8}$$

is called the Lagrange dual problem to the primal problem

$$\begin{aligned} \underset{x \in \mathcal{D}}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \\ & h_j(x) = 0 \end{aligned}$$

Karush, Kuhn, and Tucker defined conditions for a feasible optimal point $\tilde{x}$ of a given primal minimization problem.

DEFINITION 28 (*Karush-Kuhn-Tucker conditions*)

Let the convex functions (f, g, h) define a primal minimization problem. Then each optimal point $\tilde{x}$ fulfills the following conditions.

- $0 \in \partial_x \left(f(\tilde{x}) + \sum_{i=1}^{m} \lambda_i g_i(\tilde{x}) + \sum_{j=1}^{p} \mu_j h_j(\tilde{x}) \right)$ *(stationarity)*
- $\lambda_i g_i(\tilde{x}) = 0$ *(complementary slackness)*
- $g_i(\tilde{x}) \leq 0$, $h_j(\tilde{x}) = 0$, $\forall i, j$ *(primal feasibility)*
- $\lambda_i > 0 \forall i$ *(dual feasibility)*

These conditions are called Karush-Kuhn-Tucker conditions.

We will go into further detail about numerical optimization in later chapters. Especially the numerical solution and different numerical methods are discussed when needed.

2.6 Statistical Basics

This work will have some sections focusing on how distributions and random samples are compared, how noise is modeled, and confidence bands for measurements. In addition, to ensure that all readers are familiar with the basics, we need to define different distributions and interval estimates. We refer to [14] for more information about these topics. All the following definitions and topics in this section are based on random variables.

DEFINITION 29 (*Random Variable*)

A random variable $X : S \to \mathbb{R}$ is a measurable function from a sample space S into the real numbers. The sample space S consists of all possible outcomes. If the sample space S is countable, then X is said to be a discrete random variable. Otherwise, X is called a continuous random variable.

The probability that X takes on a value in a measurable set $E \subset \mathbb{R}$ is denoted as

$$P(X \in E) = P(\{w \in S | \, X(w) \in E\})$$

In tabular 2.2 we listed some examples of random variables.

DEFINITION 30 (*Cumulative Distribution Functions*)

The cumulative distribution function or cdf of a random variable X, denoted by $F_X(x)$, is defined by

$$F_X(x) = P_X(X \leq x), \ \forall x.$$

Experiment	random variable
Toss two dice	X = sum of the numbers
Toss a coin 25 times	X = number of the heads in 25 tosses
Apply different amounts of fertilizer to corn plants	X = yield per acre

Table 2.2: Examples of random variables

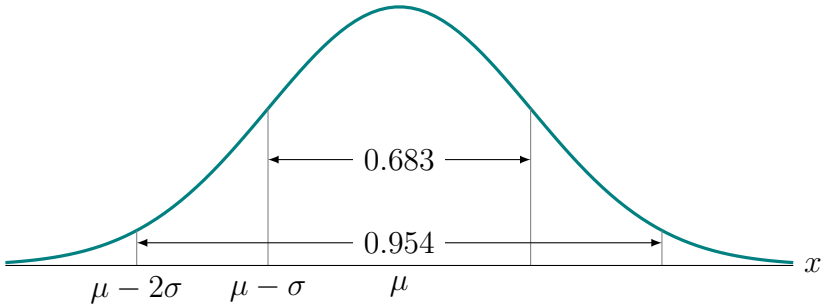

Figure 2.2: Normal distribution and its properties.

All cumulative distribution functions share the same properties, which are monoton increasing and bounded between zero and one. Besides the cumulative distribution, we are also interested in the probability of a single event. The mass function does provide us with this information and is defined with the help of the cumulative distribution function.

DEFINITION 31 (*Mass function*)

The probability mass function (pmf) of a discrete random variable X is given by

$$f_X(x) = P(X = x), \forall x.$$

The probability density function or pdf, $f_X(x)$, of a continuous random variable X is the function that satisfies

$$F_X(x) = \int_{-\infty,x} f_X(t)dt, \forall x.$$

Note that in the case of a continuous random variable, the measure of $X = x$ is zero, but $f_X(x)$ is not equal to zero.

2.6.1 Distributions

Distributions can be classified using parameterized pdfs. Various families of distributions exist and are studied. Casella and Berger provide a list of common distributions, their families, and properties in [14]. In this work we focus on distributions that are used to model noise in measurements. Of which

the most common one is the normal distribution.

DEFINITION 32 (*Normal Distribution*)

Probability density function:

$$\phi(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Cumulative distribution function

$$\Phi(\hat{x}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\hat{x}} \exp\left(\frac{-t^2}{2}\right) dt$$

where $\hat{x} := \frac{x-\mu}{\sigma}$.

A normal distribution is defined over the whole $\mathbb{R}$. It is, therefore, possible that an additive error, modeled using a normal distribution, is arbitrarily large. This arbitrarily large error contrasts with the standard assumption for inverse problems that the observed data is within a specific range of the actual data. In addition, many applications have physical boundaries on measurements which contradicts to an additive normal error model. Therefore, a truncated normal noise model is a better choice to represent the boundness of the error while approximating a normal distribution at the same time.

A truncated normal distribution is derived from a normal distribution via conditioning. It needs two additional parameters, $a, b \in \mathbb{R} \cup \pm\infty$, $a < b$, in addition to the parameter of a normal distribution.

DEFINITION 33 (*Truncated Normal Distribution*)

Suppose $X \sim N(\mu, \sigma^2)$ *has a normal distribution, then* X *conditional on* $a < X < b$ *has a truncated normal distribution with probability density function*

$$f(x;\mu,\sigma,a,b) = \frac{\phi\left(\frac{x-\mu}{\sigma};0,1\right)}{\sigma\left(\Phi\left(\frac{b-\mu}{\sigma};0,1\right) - \Phi\left(\frac{a-\mu}{\sigma};0,1\right)\right)}$$

for $a \leq x \leq b$ *and* $f = 0$ *otherwise.* ϕ *is the normal distribution probability density function and* Φ *is the cumulative distribution function.*

We will use this noise model for normal additive error in simulations to ensure the error is bounded and all methods assuming a bounded error can be applied. We also apply this model if physical bounds exist, such as the measurement must be positive.

2.6.2 Point and Interval Estimations

For a distribution a real-value parameter θ is a single point estimate. These point estimates include the mean and center of a distribution. Instead of only

estimating the parameter, a likelihood interval of this estimate can be made. This interval estimate is defined below. We follow the definition given in [14].

DEFINITION 34 (*Interval Estimate*)

An interval estimate of a real-value parameter θ is any pair of functions $L(x_1, \ldots, x_n)$ and $U(x_1, \ldots, x_n)$, of a sample that satisfy $L(x) \leq \theta \leq U(x)$ for all $x \in X$. If $X = x$ is observed, the inference $L(x) \leq \theta \leq U(x)$ is made. The random interval $[L(X),\ U(X)]$ is called an interval estimator.

We can say that an interval estimate provides extra information about the point estimate. It also includes information about the accuracy of this estimate. In that sense, it contains more information and is more meaningful.

A typical example of a point estimate is an observation's mean and expected value given a set of random samples. The corresponding interval estimates the confidence interval of this estimated mean based on the number of samples and assumed distribution. We will refer to these interval estimates from multiple random samples when comparing our method, which includes tolerance in discrepancy with other methods.

2.6.3 Comparing Distributions

We have various methods to choose from for comparing different distributions and calculating their similarities. This section will present a few common methods to calculate the similarities between two distributions. We will compare distributions in the application of micro milling, where we characterize surfaces of micro parts.

The *Earth Movers distance* or *Wasserstein metric* for histograms is defined as the solution to the minimum transportation problem between the two distributions. The EMD can be calculated using the following iterative method for two histograms.

$$\begin{aligned} EMD_0 &= 0 \\ EMD_{k+1} &= f_X(x_i) + EMD_k - f_Y(x_i) \\ d(X,Y) &= \sum_i |EMD_i| \end{aligned}$$

Based on the Pearson's chi squared test statistic

$$\chi^2(x,y) = \sum((x_i - y_i)^2/x_i)$$

the χ^2-distance is defined as

$$d(X,Y) = \frac{1}{2}\sum_i \frac{(f_X(x_i) - f_Y(x_i))^2}{f_X(x_i) + f_Y(x_i)}$$

for histograms and as

$$d(X,Y) = \frac{1}{2}\int_{\Omega} \frac{(f_X(t) - f_Y(t))^2}{f_X(t) + f_Y(t)} dt$$

for probability functions. The χ^2-distance is a true metric since it is symmetric with respect to x and y. Contrary to the Earth Movers distance, the χ^2-distance can be computed directly without using any optimization problem.

Another common metric is the *Mahalanobis distance* defined as

$$d(X,Y) = \sqrt{(f_X(x) - f_y(x))^T S^{-1} (f_X(x) - f_Y(x))}$$

with S the non-singular covariance matrix matrix and $x \in \mathbb{R}^n$ a vector. For $S = I$ the Mahalanobis distance is equal to the euclidean distance.

3 Tolerances in Hilbert Spaces

Using a-priori information and assumptions on the true solutions when solving inverse problems, we can tackle the ill-posed nature of operators and stabilize the source reconstruction of observed measurements. Therefore, regularization is achieved by exploiting features and structures of the parameter space or said a-priori information about the true parameter. For example, in classic Tikhonov functionals, the norm or smoothness of the reconstructed parameter is used to stabilize the problem [50]. Another example is sparsity, where we assume that the true parameter is sparse in a particular basis [36]. However, not only assumptions and information about the true parameter are available. In some cases, additional information about the measurement may be available. So far, this knowledge is rarely used in Inverse Problems besides choosing an appropriate norm for the deviation in the data space for Tikhonov methods. In this chapter, we introduce the idea of exploiting structures within the measurement and data space to stabilize and enhance the reconstruction process in ill-posed Inverse Problems. This idea originates from statistical methods, such as Support Vector Machines and noise modeling. We focus on Support Vector Machines and the idea of using insensitive distance measurements to introduce tolerances in the data discrepancy for Tikhonov functionals in Hilbert spaces.

3.1 Classification and Regression with Support Vectors

For classification and regression a function $f : X \to Y$, where $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^m$, is derived from a data set with n data points

$$\left\{(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}^m \;\middle|\; i = 1, \ldots, n\right\}.$$

We refer to this set as the training set. The function f approximates the assumed underlying connection of x_i to y_i and is used later to predict an unknown output y of a new input x. If the output of f is discrete and finite, f

is said to be a classification. If it is continuous, it is a regression. One method to obtain f from known data (x_i, y_i) is using support vector machines (SVMs). Nowadays, they are overshadowed by neural networks and deep learning but are still applied to specialized classification and regression methods.We refer to it as support vector regression (SVR) in the latter case.

We will focus on the support vector methods since they are closely related to discrepancy terms with tolerances for inverse problems. For further information on classification and regression methods, as well as support vector methods, we refer the reader to [70] and [77].

3.1.1 SVMs for Classification

Support Vector Machines (SVMs) for classification were first introduced by Vladimir N. Vapnik and Alexey Ya Chervonenkis in 1963. The idea is that a hyperplane that maximizes the distance between each class can best separate two classes. Such a hyperplane is unique if a separation by a hyperplane is possible, and it minimizes the out-of-sample error. This error measures how well a classifier works on unlearned data points (see [69] chapter 5,7,12).

To understand how SVMs are used for classification, we need to look at hyperplanes in general. They are defined as follows:

DEFINITION 35 (*Hyperplane*)

Let $x \in H$ and $d \in \mathbb{R}$, where H is some Hilbert space. The set

$$L = \{y \in H \mid \langle x, y\rangle + d = 0\}$$

is a hyperplane in H defined by x and d.

First, the output Y is restricted to the set $\{-1, 1\}$ since only two classes are assumed to exist. Secondly, we say a hyperplane

$$L_{w,\beta_0} := \{x \in X \mid x \in \langle w, x\rangle + \beta_0 = 0\},\ w \in X,\ \beta_0 \in \mathbb{R}$$

is separating the data if

$$f(x) := \operatorname{sgn}(\langle w, x_i\rangle + \beta_0) = y_i, \quad \text{for } i = 1, \ldots, n.$$

In the case that at least one such hyperplane exists we say that the data is separable. This means all data belonging to one class are at one side of the hyperplane while all data of the other class are at the other side. An example of a separable data set is shown in figure 3.1.

From vector algebra we know that the signed distance of a point to the hyperplane L_{w,β_0} is

$$dist(x, L) = \frac{1}{\|w\|}(\langle w, x\rangle + \beta_0).$$

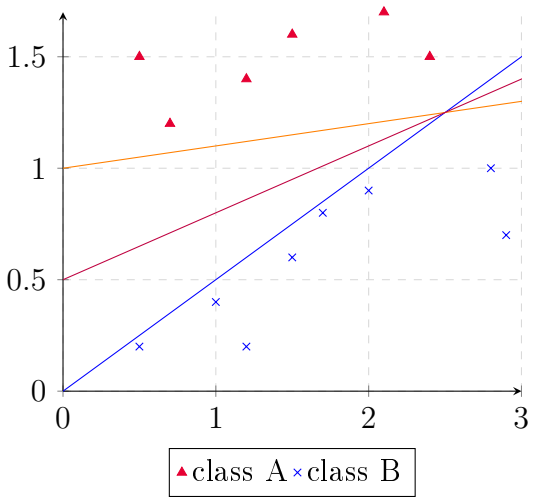

Figure 3.1: A toy example of a separable data set. The two classes are marked with red triangles and blue stars. The three solid lines are different separating hyperplanes.

Thus the classifier f based on L_{w,β_0} can be defined as

$$f(x) = \operatorname{sgn}\left(\langle w, x\rangle + \beta_0\right).$$

By this definition we ensure that the classifier maps correctly onto $Y = \{-1, 1\}$. If L_{w,β_0} is indeed a separating hyperplane, f classifies all data in the training set correctly. In general, separating hyperplanes are not unique for separable data sets, which leads to the question of which one is the best separating hyperplane. As mentioned in the beginning of the section, one logical choice is the hyperplane that maximizes the distance to both classes. To find this hyperplane we consider the optimization problem

$$\begin{aligned} &\text{maximize} && M \in \mathbb{R}^+ \\ &\text{subject to} && \frac{y_i}{\|w\|}(\langle w, x_i\rangle + \beta_0) \geq M,\ i = 1, \ldots, n \end{aligned} \tag{3.1}$$

whose solution provides the desired hyperplane L_{w,β_0} where all points have at least distance M from the hyperplane. The condition

$$\frac{y_i}{\|w\|}(\langle w, x_i\rangle + \beta_0)$$

ensures that all points are classified correctly to have a positive value. A missclassified point, x_i, will have the opposite sign of y_i and thus $\frac{y_i}{\|w\|}(\langle w, x_i\rangle + \beta_0) \leq 0$.

The set of all points with distance M to the hyperplane L_{w,β_0} is called margin. This optimization problem does not have a unique solution but can be transformed to a convex optimization problem whose solution is unique.

Let $\alpha \in \mathbb{R}^+$ be arbitrary, then

$$\begin{aligned}\frac{1}{\|\alpha w\|}(\langle \alpha w, x\rangle + \alpha\beta_0) &= \frac{\alpha}{\alpha\|w\|}(\langle w, x\rangle + \beta_0) \\ &= \frac{1}{\|w\|}(\langle w, x\rangle + \beta_0).\end{aligned}$$

If a pair of w and β_0 maximize M, the positively scaled w and β_0 maximize M as well. Let $M := \frac{1}{\|w\|}$ we can transform the problem (3.1) to the convex optimization problem

$$\begin{aligned}&\text{minimize} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to} \quad y_i(\langle w, x_i\rangle + \beta_0) \geq 1,\ i = 1, \ldots, n.\end{aligned} \tag{3.2}$$

This transformation is possible because of the property that continuous convex transformations do not change the *argmin*. Moreover, because the optimization problem for finding the desired hyperplane has been converted to a convex one, the SVM has good properties for numerical implementation. For a non-separable data set the optimization problem does not have a feasible solution, i.e. there exist no hyperplane that fulfills the constrain in (3.2).

One way to find a minimizer for (3.2) is to alter the restrictions by allowing a few points to be on the wrong side of the hyperplane and/or within the margin. We define the slack variable $\varepsilon = \{\varepsilon_1, \ldots, \varepsilon_n\}$, and use it to modify the constrains in (3.1). There are two options, tolerating the actual distance

$$\frac{y_i}{\|w\|}(\langle w, x_i\rangle + \beta_0) \geq M - \varepsilon_i \tag{3.3}$$

or the relative distance

$$\frac{y_i}{\|w\|}(\langle w, x_i\rangle + \beta_0) \geq M(1 - \varepsilon_i). \tag{3.4}$$

In both cases we restrict $\varepsilon > 0$ and $\sum_{i=1}^n \varepsilon_i \leq c$ with $c \in \mathbb{R}_+$ constant. The restrictions (3.3) and (3.4) lead to different solutions. Since (3.3) leads to a non-convex optimization problem and (3.4) does not, the second choice is the standard choice for SVMs. Please note that we did not bound $\varepsilon_i \leq 1$, which means that a point can be on the wrong side of the hyperplane. Similarly to the previous case with separable data we can rephrase (3.1) with the new restriction (3.2) to

$$\begin{aligned}&\text{minimize} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to} \quad y_i(\langle w, x_i\rangle + \beta_0) \geq 1 - \varepsilon_i,\ i = 1, \ldots, n \\ &\qquad\qquad \varepsilon_i \geq 0,\ \sum_{i=1}^n \varepsilon_i \leq c\end{aligned} \tag{3.5}$$

where $c \geq 0$ is a predefined constant. Note that for large enough c a minimizer exists even for non-separable data.

The Lagrangian of (3.5) and its optimality conditions yield

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

with $\alpha_i \in \mathbb{R}$. Since for all $\alpha_i = 0$, x_i are not within the margin of the minimizer, the solution w is a linear combination of training samples within the margin. The x_i within the margin are referred to as support vectors and give the support vector machine its name.

So far, we only described the primal problem to find a separating hyperplane. By transforming the minimization into its dual via Lagrangian multipliers, we can calculate the support vector machine by maximizing

$$W(\Lambda) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_j \alpha_i y_i y_j \langle x_j, x_i \rangle, \tag{3.6}$$

where $\Lambda = (\alpha_1, \ldots, \alpha_n)$. The maximizer for equation (3.6) is also a minimizer for its primal (3.5). While the primal problem can be solved efficiently with numerical methods, the dual problem has nice mathematical properties that are useful when we look at non-linear SVMs in this chapter.

3.1.2 SVMs for Regression

So far, the introduced SVMs are used to classify data into two groups. We can extend the approach to multiple classes via assembling methods such as "one vs. all" approaches. However, in this thesis, we are more interested in adopting support vectors for regression. In regression, the output of the underlying connection between x_i and y_i is continuous. Therefore f can not separate some data sets. Instead we search for a function f such that $f(x_i) = y_i$ for all $i = 1, \cdots, n$. Since such an f may not exist, we solve a minimization problem

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

instead. Here a minimization in the 2-norm was chosen. Other distance measure can also be considered. Since it is possible that the same input $x_i = x_j$ and $i \neq j$ have different output $y_i \neq y_j$, a function that perfectly maps x_i onto y_i may not exist, which is not an exception but occurs regularly. In addition, f should have a small out-of-sample error, which means that overfitting the training data should be avoided. For support vector regression, small data variations are being tolerated by applying an $\varepsilon-$*insensitive loss function*

$$|f(x) - y|_\varepsilon := \max\{0, |f(x) - y| - \varepsilon\},$$

where $\varepsilon \geq 0$, as introduced by Vapnik [81]. In figure 3.2 an illustration of this function and its application for regression is given.

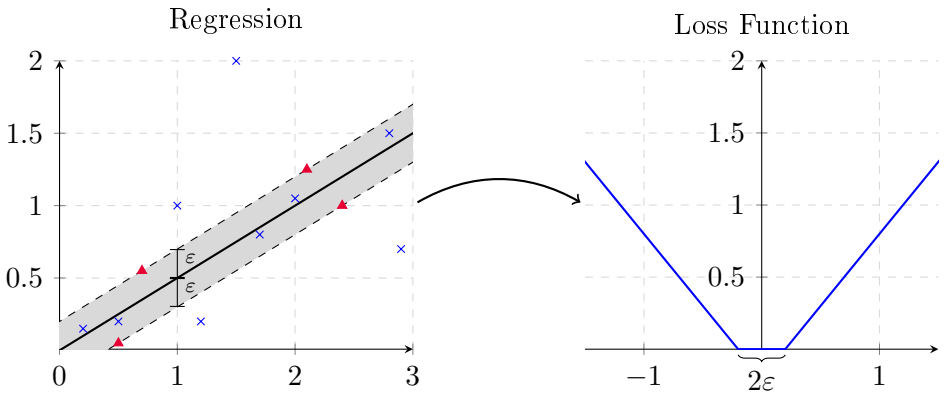

Figure 3.2: The left panel shows the regression obtained using SVR and the least square method. In red and blue are the data points. The red triangle data points are the support vectors. All points within the grey area are zero in the loss function for the SVR. The right panel shows the used loss function.

To find a regression function f, a minimization with introduced insensitive loss function on the training data

$$\tilde{f} := \operatorname*{argmin}_f \sum_{i=1}^{n} |f(x_i) - y_i|_\varepsilon \tag{3.7}$$

is applied. As in the case of SVMs for classification, this minimizer is not unique and the minimization is numerically unstable. To solve this problem Vapnik proposes to add a weighted penalty term on the function f. This added penalty results in

$$\sum_{i=1}^{n} |f(x_i) - y_i|_\varepsilon + \alpha \|f\|_2^2$$

with $\alpha > 0$ as the weight. The derived functional is to be very similar to a Tikhonov functional

$$J_{\delta,\alpha}^{p,q}(u) = \frac{1}{p} \left\| F(u) - v^\delta \right\|_{L_p}^p + \alpha \|u\|_q^q$$

with $p = 1$ and $q = 2$. Finding a minimizer of this functional uses a similar approach to solve 3.1. The support vector regression with $\varepsilon = 0$ is known as rigid regression, a type of linear regression used to avoid overfitting on the training set.

3.2 Nonlinear SVR

Support vector machines perform a linear classification or regression. They are not linear models since the model coefficients are non-linear, but the fitted

regression or classification function is linear. Section 2 provides a short introduction to linear models in regression to clarify this statement further. In order to find non-linear decision models, the so-called kernel trick is applied [69]. All input data x_i is transformed through a kernel operation σ into a feature space. This feature space is often of higher dimension than the original input space.

Let us start with an example for understanding the principle idea. Figure 3.3 shows an example for a two-dimensional toy data set. The data represents samples from two classes where the first class, labeled *Group A*, is centered and dense around the origin $(0, 0)$ with a maximum distance $\|x\| < 1$. The second class, labeled *Group B*, is centered around the same origin $(0, 0)$ but with a minimum distance of $\|x\| > 1$. A single hyperplane cannot separate these two sets. Instead, the two groups can be separated by an ellipse which is not a linear decision boundary. To find a proper separating hyperplane, we can transform all data into a new feature space. We transform the data with the kernel

$$\Sigma(x, y) = \left(x, y, x^2 + y^2\right)$$

into a new feature space where the hyperplane

$$\langle n, (x, y, z)^T \rangle - 1 = 0$$

with $n = (1, 0, 0)^T$ and $z = x^2 + y^2$ forms a suitable decision boundary in the feature space.

We derived this solution from knowledge about the underlying distribution of the classes. The resulting decision boundary might vary if we calculate an SVM classifier from random samples.

Instead of transforming all data into the feature space and then solving the linear problem, the support vectors and corresponding decision function can be calculated directly by solving the dual problem. This property is beneficial if the feature space is much larger than the original space. Remember that the number of variables for the primal problem is equal to the dimension of the input space. In contrast, the number of variables for the dual problem is equivalent to the size of the training set. In addition, we do not need to actually transform all samples but instead use a kernel function directly in the dual problem.

Table 3.1 shows the most commonly used kernels and their inner product. The correct kernel and its parameters must be given a-priori or can be found with a secondary optimization and cross-validation. Different approaches for kernel and parameter selection have been introduced over the years. We will provide an insight on such methods in chapter 4 where we consider the numerical solution of inverse problems.

3.3 Definition of Tolerance in Inverse Problems

When dealing with tolerances, a proper understanding of tolerances itself is essential. Different fields of science and application have varying definitions

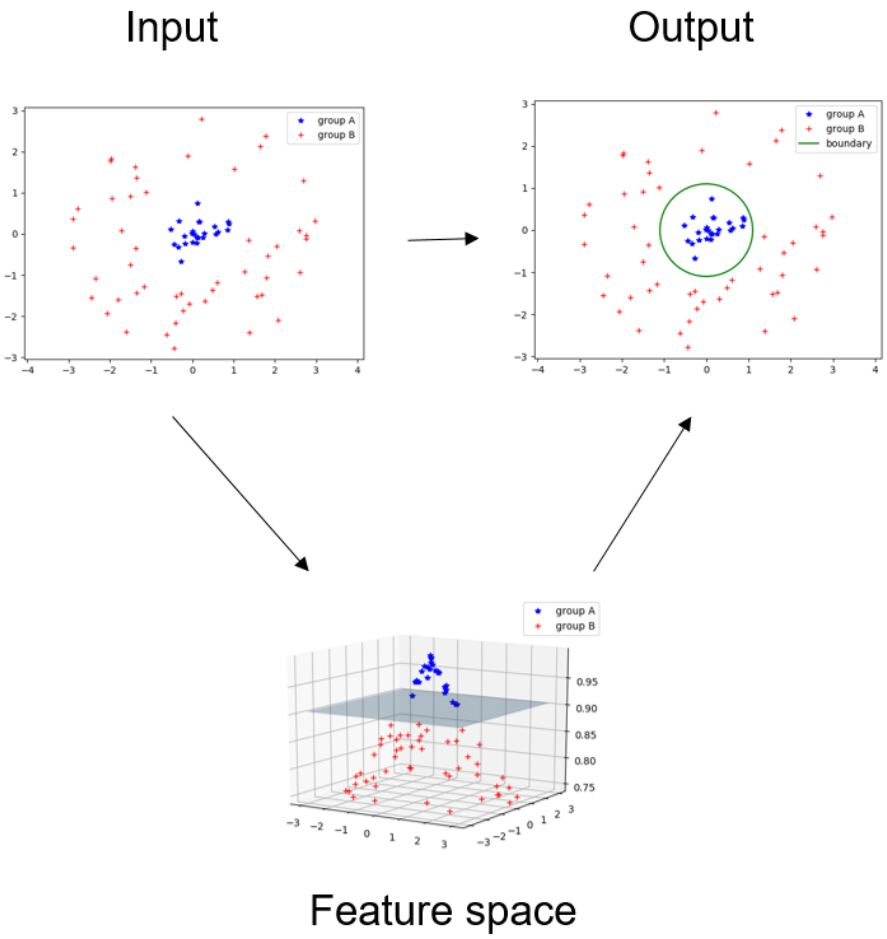

Figure 3.3: Example of non-linear SVM. The applied kernel separates the data in the feature space through a hyperplane. Projecting the separating hyperplane back into the feature space yields a non-linear decision boundary.

Kernel	Inner Product	Parameters
Linear	$x_j^T x_i$	none
Polynomial	$(x_j^T x_i + c)^d$	$c \in \mathbb{R},\ d \in \mathbb{N}$
Gaussian	$\exp\left(\frac{1}{2\sigma^2}\|x_j - x_i\|^2\right)$	$\sigma \in \mathbb{R}_+$
Sigmoid	$\tanh\left(s \cdot x_j^T x_i + c\right)$	$s \in \mathbb{R},\ c \in \mathbb{R}$
Rational Quadratic	$1 - \frac{\|x_j - x_i\|^2}{\|x_j - x_i\|^2 + c}$	$c \in \mathbb{R}$
Inverse Multiquadratic	$\left(\sqrt{\|x_j - x_i\|^2 + c^2}\right)^{-1}$	$c \in \mathbb{R}$

Table 3.1: Examples of kernels and their inner product.

for tolerances. It is, therefore, necessary to establish a definition of tolerances in this thesis to achieve a common understanding for the reader. In this thesis, we focus on tolerances in engineering and mathematics.

For a single value $z \geq 0$, the two numbers $x, y \in \mathbb{R}$ are within a tolerance of z if

$$dist(x, y) \leq z,$$

where $dist : \mathbb{R} \times \mathbb{R} \Rightarrow \mathbb{R}^+$ is a distance function. This formulation shows that two things are important for a tolerance: how to define the distance function of two elements and the value or size of the tolerance. Together they define a tolerance around a fixed element x_0. It is important to note that this definition does not allow negative tolerances. We can extend this definition to more complex elements. So far, x and y are real numbers. In order to define a tolerance on a more general case, x and y need to be elements of the same metric space (X, d). Then for a single value $z \geq 0$, the two elements $x, y \in X$ are within a tolerance of z if

$$q(x, y) \leq z,$$

where $q : X \times X \to \mathbb{R}$ is a pseudo metric (also known as premetric in some literature).

We summarize the above ideas into one formal definition, which we will use in the remainder of this thesis.

DEFINITION 36 (*Tolerance*)

For $f \in V$ Hilbert space the area of tolerance $\Omega_{f,\varepsilon} \subset V$ is a closed set in V and contains at least f. All elements $g \in \Omega_{f,\varepsilon}$ have a maximum distance of ε to f, i.e.

$$d(g, f) \leq \varepsilon, \forall g \in \Omega_{f,\varepsilon}.$$

It is important to note that by our definition, two elements can have a distance of less than ε but still not be within tolerance to each other, i.e.,

$$d(g, f) \leq \varepsilon \not\Rightarrow g \in \Omega_{f,\varepsilon}$$

A simple example illustrating different tolerances and their application can be found in shape deviations. In micro forming processes, it is oftentimes needed to compare shapes to sort good parts from defective parts. Since deviations of manufactured components cannot be avoided, a tolerance for accepting good parts is necessary. In this example, the opening of a cup is under consideration. A perfect cup will have a circle-shaped opening with a radius $R = 5mm$, forming a perfect circle. In figure 3.4 the deviation of two measured cups is shown in red and blue. Two possible ways to describe a tolerance for this case are the total deviation over the whole circumference,

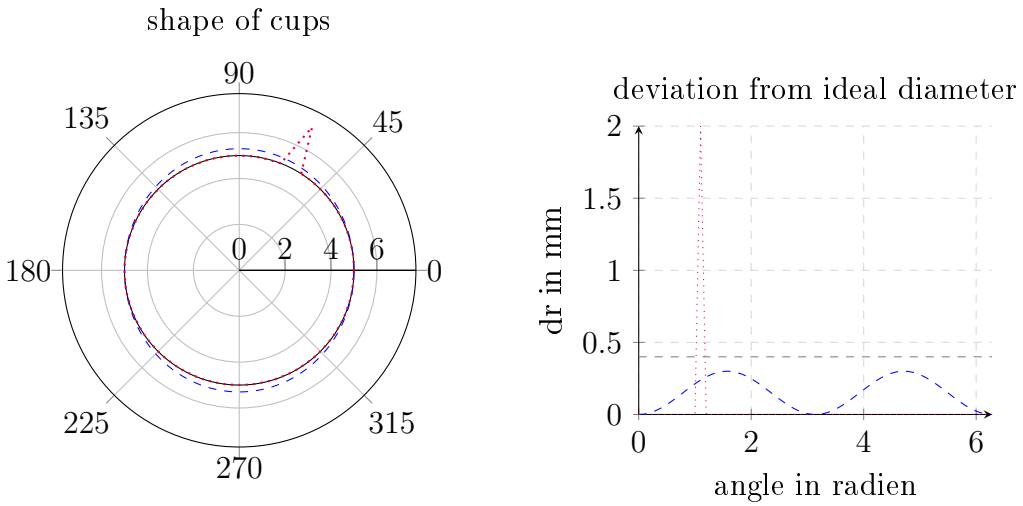

Figure 3.4: Example of difference in point wise and overall tolerance for shape deviations of micro cups.

q_1, or a point-wise tolerance for each angle, q_2.

$$q_1(x,y) = \int_0^{2\pi} (x(\varphi) - y(\varphi))\ d\varphi$$
$$q_2(x,y) = \int_0^{2\pi} \max\{0,\ |x(\varphi) - y(\varphi)| - \varepsilon\}\ d\varphi$$

In the example the red cup has a deviation of $q_1(x, y_{blue}) = 0.5$ which is less than the blue cup with a deviation of $q_1(x, y_{red}) = 0.931$. The deviation with tolerance is $q_2(x, y_{blue}) = 0.281$ and $q_1(x, y_{red}) = 0$. For q_2, the blue cup is totally within tolerance and has a lower distance to the ideal shape than the red cup. q_1 measures the overall deviation of the shape regardless of any tolerance whereas q_2 considers a point-wise tolerance. This point-wise tolerance can be interpreted as a tolerance band around the ideal shape, and as long as a cup shape is within this tolerance band, it is accepted. Thus the red cup has a distance of 0 in the measurement q_2 since it is completely within the tolerance band. It is important to note that there exists no threshold on q_1, which accepts only shapes within the tolerance band. The red and blue cups are an example of this. The blue cup has a lower distance in q_1 and would always be within tolerance to the ideal shape if the red cup is within the tolerance, considering q_1. Therefore, the distance q_2 defines a different tolerance.

3.4 Mathematical Framework

This section provides a mathematical framework for Tikhonov functionals with tolerances in the discrepancy terms. This includes existence of a minimizer, stability in δ and ε and convergence for $\delta \to 0$ and $\varepsilon \to 0$.

The following assumptions on the operator and functional are made:

Assumption 1

1. *The operator F is weakly sequentially closed with respect to the weak topologies on $L_p(\Omega)$.*
2. *The functional $\mathcal{R} : U \to \mathbb{R}_+$ is coercive.*
3. *The intersection of the domains of F and $\mathcal{R}$ is non-empty, i.e.*

$$\mathcal{D} := \mathcal{D}(F) \cap \mathcal{D}(\mathcal{R}) \neq \emptyset$$

and the following notations will be applied:

- An element $u^\dagger \in \mathcal{D}$ denotes a $\mathcal{R}_q$-minimizing solution i.e.

$$\mathcal{R}(u^\dagger) = \min \{\mathcal{R}_q(u) |\ F(u) = f\} < \infty$$

- $u^\delta_{\alpha,\varepsilon}$ denotes a minimizer of the functional $J^{p,q}_{\delta,\alpha,\varepsilon}$ in $\mathcal{U}$:

$$u^\delta_{\alpha,\varepsilon} = \operatorname{argmin} \left\{ J^{p,q}_{\delta,\alpha,\varepsilon}(u) |\, u \in \mathcal{U} \right\}$$

Before we can show existence and stability, we prove a lemma from [28] for our setting with tolerances.

First, we define the functionals

$$J^{p,q}_{\alpha,\varepsilon,v_k}(u) := \frac{1}{p} \|F(u) - v_k\|^p_{L_p,\varepsilon} + \alpha \mathcal{R}_q(u),$$

which is the same functional as $J^{p,q}_{\delta,\alpha,\varepsilon}$ for $v_k = v^\delta$, and

$$J^{p,q}_{\alpha,v_k}(u) := \frac{1}{p} \|F(u) - v_k\|^p_{L_p} + \alpha \mathcal{R}_q(u)$$

for the special case of $\varepsilon = 0$ which is the Tikhonov functional without ε-insensitive distance.

LEMMA 37

Let $(u_k)_{k\in\mathbb{N}} \subset dom(F)$ and $(v_k)_{k\in\mathbb{N}} \subset V$. Assume that the sequence $(v_k)_{k\in\mathbb{N}}$ is bounded in V and that there exist $\alpha > 0$ and $M > 0$ such that $J^{p,q}_{\alpha,\varepsilon,v_k}(u_k) < M$ for all $k \in \mathbb{N}$. Then there exist $u \in dom(F)$ and a subsequence $(u_{k_j})_{j\in\mathbb{N}}$ such that $u_{k_j} \rightharpoonup u$ and $F(u_{k_j}) \rightharpoonup F(u)$.

Proof: The coercivity of $\mathcal{R}_q$ and the estimate $J^{p,q}_{\alpha,\varepsilon,v_k}(u_k) \geq \alpha \mathcal{R}_q(u_k)$ imply that the sequence $(u_k)_{k\in\mathbb{N}}$ is bounded in U. To prove this we assume that the

sequence (u_k) is unbounded. Therefore a subsequence $(u_{k_l})_{l\in\mathbb{N}}$ exist such that $\|u_{k_l}\|_U \to \infty$ for $j \to \infty$. From the inequality

$$0 \leq \frac{\alpha \mathcal{R}_q(u_{k_l})}{\|u_{k_l}\|_U} \leq \frac{M}{\|u_{k_l}\|_U}$$

follows

$$\lim_{l\to\infty} \frac{\alpha \mathcal{R}_q(u_{k_l})}{\|u_{k_l}\|_U} = 0$$

and therefore $\mathcal{R}_q$ is not coercive. This is a contradiction and thus (u_k) must be bounded.

Since the sequence $(v_k)_{k\in\mathbb{N}}$ is bounded, also the sequence $(F(u_k))_{k\in\mathbb{N}}$ is bounded in V. To see this we apply the second inequality from remark 48 to have

$$\begin{aligned}\|F(u_k) - v_k\|_{L_p}^p &\leq J_{\alpha,v_k}^{p,g}(u_k) \\ &\leq J_{\alpha,\varepsilon,v_k}^{p,g}(u_k) + \|\varepsilon\|_{L_p}^p \\ &\leq M + \|\varepsilon\|_{L_p}^p\end{aligned}$$

and this is a bound for $\|F(u_k) - v_k\|_{L_p}^p$. Thus the sequence $(F(u_k))_{k\in\mathbb{N}}$ has to be bounded since $(v_k)_{k\in\mathbb{N}}$ is bounded.

Therefore there exist a subsequence $(u_{k_j})_{j\in\mathbb{N}}$ and $u \in U$, $v \in V$, such that $(u_{k_j})_{j\in\mathbb{N}}$ weakly converges to u and $(F(u_{k_j}))_{j\in\mathbb{N}}$ weakly converges to y. Since F is weakly sequentially closed, it follows that $u \in \mathrm{dom}(F)$ and $F(u) = y$. ■

With the help of this Lemma, we can show the existence of a minimizer and stability.

LEMMA 38 (*Existence of minimizer*)

Assume that $\delta > 0$ and $v^\delta \in V$. Let $\mathcal{D} := \mathcal{D}(F) \cap \mathcal{D}(\mathcal{R}) \neq \emptyset$ and let F satisfy the assumptions made at the beginning of this section in assumption 1. Then $J_{\delta,\alpha,\varepsilon}^{p,q}$ attains a minimizer.

Proof: Let $(u_k)_{k\in\mathbb{N}} \subset \mathcal{U}$ be a sequence that satisfies

$$\lim_{k\to\infty} J_{\delta,\alpha,\varepsilon}^{p,q}(u_k) = \inf\left\{ J_{\delta,\alpha,\varepsilon}^{p,q}(u) \,\middle|\, u \in \mathcal{U} \right\}.$$

For $q \geq 2$ the functional $J_{\delta,\alpha,\varepsilon}^{p,q}$ is coercive and hence (u_k) is bounded. In particular $(u_k)_{k\in\mathbb{N}}$ admits a subsequence, which we again denote by $(u_k)_{k\in\mathbb{N}}$, that weakly converges to some $\tilde{u} \in \mathcal{U}$ such that $F(u_k) \rightharpoonup F(\tilde{u})$ (see, [28, Lemma 4]). Since $\mathcal{R}$ is lower semi continuous as the sum of non-negative convex and weakly continuous functionals, we have

$$\mathcal{R}(\tilde{u}) \leq \liminf_{k\to\infty} \mathcal{R}(u_k). \tag{3.8}$$

Moreover, since $J^{p,q}_{\delta,\alpha,\varepsilon}$ is weakly lower semi continuous due to the weak sequential closedness of F and the weak lower semicontinuity of the norm we have

$$J^{p,q}_{\delta,\alpha,\varepsilon}(\tilde{u}) = \left\|F(\tilde{u}) - v^\delta\right\|^p_\varepsilon + \alpha\mathcal{R}_q(\tilde{u}) \tag{3.9}$$

$$\leq \liminf_{k\to\infty}\left(\left\|F(u_k) - y^\delta\right\|^p_\varepsilon + \alpha\mathcal{R}_q(u_k)\right). \tag{3.10}$$

We also have

$$\inf\left\{J^{p,q}_{\delta,\alpha,\varepsilon}(u)\middle|\, u\in\mathcal{U}\right\} \leq J^{p,q}_{\delta,\alpha,\varepsilon}(\tilde{u}),$$

which completes the proof. ■

LEMMA 39 (*Stability for δ*)

Let $(v_k)_{k\in\mathbb{N}} \subset V$ converge to v^δ and let

$$u_k \in \operatorname{argmin}\left\{J^{p,q}_{\alpha,v_k,\varepsilon}(u)\middle|\, u\in\mathcal{D}\right\}.$$

Then there exist a subsequence $(u_{k_j})_{j\in\mathbb{N}}$ and a minimizer $u^\delta_{\alpha,\varepsilon}$ of $J^{p,q}_{\alpha,0,\varepsilon}$ such that $\mathcal{R}_q(u^\delta_{\alpha,\varepsilon} - u_{k_j}) \to 0$. If the minimizer $u^\delta_{\alpha,\varepsilon}$ is unique, then $(u_{k_j})_{j\in\mathbb{N}}$ converges to the minimizer $u^\delta_{\alpha,\varepsilon}$ with respect to $\mathcal{R}_q$.

Proof: This proof has the same structure as the proof in [28, Proposition 4] and follows its main idea, which is applying lemma 37. Note that ε and α are fixed and do not change with k in lemma 39.

We have two sequences $(v_k)_{k\in\mathbb{R}}$ and $(u_k)_{k\in\mathbb{R}}$. Since the sequence $(v_k)_{k\in\mathbb{R}}$ is a convergent sequence, it is bounded as well. Each $u_k \in \operatorname{argmin} J^{p,q}_{\alpha,v_k,\varepsilon}$ and thus

$$J^{p,q}_{\alpha,v_k,\varepsilon}(u^\delta_{\alpha,\varepsilon}) \geq J^{p,q}_{\alpha,v_k,\varepsilon}(u_k) \geq 0.$$

The left side has an upper bound of

$$\begin{aligned} J^{p,q}_{\alpha,v_k,\varepsilon}(u^\delta_{\alpha,\varepsilon}) &= \|F(u^\delta_{\alpha,\varepsilon}) - v_k\|^p_{V,\varepsilon} + \alpha\mathcal{R}_q(u^\delta_{\alpha,\varepsilon}) \\ &\leq 2^{p-1}\|F(u^\delta_{\alpha,\varepsilon}) - v^\delta\|^p_{V,\varepsilon} + 2^{p-1}\|v_k - v^\delta\|^p_{V,\varepsilon} + \alpha\mathcal{R}_q(u^\delta_{\alpha,\varepsilon}) \end{aligned} \tag{3.11}$$

for all k since $(v_k)_{k\in\mathbb{R}}$ is bounded and therefore $J^{p,q}_{\alpha,v_k,\varepsilon}(u_k)$ is bounded. All requirements for lemma 37 are met and $(u_k)_{k\in\mathbb{R}}$ has a subsequence $(u_{k_j})_{j\in\mathbb{R}}$ weakly convergent to some $u \in \mathcal{D}(F)$ such that $F(u_k) \rightharpoonup F(u)$. Analog to the proof of existence we have that

$$J^{p,q}_{\delta,\alpha,\varepsilon}(u) \leq \liminf_j J^{p,q}_{\alpha,v_{k_j},\varepsilon}(u_{k_j})$$

since $v_k \to v^\delta$.

Additionally, for all $w \in \mathcal{D}(F)$

$$\begin{aligned} J^{p,q}_{\delta,\alpha,\varepsilon}(w) &= \lim_k J^{p,q}_{\alpha,v_k,\varepsilon}(w) \\ &\geq \liminf_k J^{p,q}_{\alpha,v_k,\varepsilon}(u_k). \end{aligned}$$

Thus $u = u^{\delta}_{\alpha,\varepsilon}$ is a minimizer of $J^{p,q}_{\delta,\alpha,\varepsilon}$.

From the inequality (3.11) we also see that $J^{p,q}_{\delta,\alpha,\varepsilon}(u_{k_j}) \to J^{p,q}_{\delta,\alpha,\varepsilon}(u)$. Both $\|\cdot\|^p_{U,\varepsilon}$ and $\mathcal{R}_q$ are weakly sequentially lower semi-continuous, which implies that $\mathcal{R}_q(u_{k_j}) \to \mathcal{R}_q(u)$. By the assumptions made on $\mathcal{R}_q$, the subsequence $(u_{k_j})_{j\in\mathbb{N}}$ converges with respect to $\mathcal{R}_q$.

In case that the minimizer is $u^{\delta}_{\alpha,\varepsilon}$ is unique, the convergence of the original sequence $(u_k)_{k\in\mathbb{N}}$ to $u^{\delta}_{\alpha,\varepsilon}$ follows from a subsequence argument. ■

LEMMA 40 (*Stability for ε*)

Let $(\varepsilon_k)_k \subset \mathbb{R}$ converge to 0 and let

$$u_k \in \operatorname{argmin}\left\{ J^{p,q}_{\alpha,v^{\delta},\varepsilon_k}(u) \;\middle|\; u \in \mathcal{D} \right\}.$$

Then there exists a subsequence $(\varepsilon_{k_j})_{j\in\mathbb{N}}$ and a minimizer u^{δ}_{α} of $J^{p,q}_{\delta,\alpha,\varepsilon}$ such that $\mathcal{R}_q(u^{\delta}_{\alpha} - u_{k_j}) \to 0$. If the minimizer u^{δ}_{α} is unique, then $(u_{k_j})_{j\in\mathbb{N}}$ converges to the minimizer $u^{\delta}_{\alpha,\varepsilon}$ with respect to $\mathcal{R}_q$.

Proof: We only need to show that lemma 37 can be applied. The rest of the proof is then analog to the proof of lemma 39 shown above.

We need two sequences $(u_k)_{k\in\mathbb{N}}$ and $(v_k)_{k\in\mathbb{N}}$. The first sequence $(u_k)_{k\in\mathbb{N}}$ is defined in the lemma and the second sequence is set to be $v_k = v^{\delta}$ for all $k \in \mathbb{N}$, which is a bounded sequence. First we note that

$$J^{p,q}_{\delta,\alpha,\varepsilon_k}(u_k) \leq J^{p,q}_{\delta,\alpha,\varepsilon_k}(u)$$

since u_k is a minimizer. From the additional inequality

$$J^{p,q}_{\delta,\alpha,\varepsilon_k}(u) \leq \|F(u) - v^{\delta}\|^p_V + \alpha\mathcal{R}_q(u)$$

it follows that $J^{p,q}_{\delta,\alpha,\varepsilon_k}(u_k)$ must be bounded. ■

LEMMA 41 (*convergence*)

Assume that a there exists a $u \in \mathcal{D}(F)$ such that $F(u) = v$. Let $(\delta_k)_{k\in\mathbb{N}} \subset \mathbb{R}$ be a convergent sequence to 0 and let $v_k \in V$ satisfy $\|v_k - v\|_V$

Proof: We follow the proof of Proposition 7 in [28]. Let $u \in \mathcal{D}(\mathcal{R}_q)$ be a $\mathcal{R}_q$-minimum solution. By the definition of u_k we have

$$\begin{aligned} J_{\alpha_k,\varepsilon_k,v_k}(u_k) &\leq \|F(u) - v_k\|^p_{p,\varepsilon} + \alpha_k\mathcal{R}_q(u) \\ &\leq 2^{p-1}(\delta^p_k + \|\varepsilon_k\|^p_p) + \alpha_k\mathcal{R}_q(u). \end{aligned}$$

This yields $\limsup \mathcal{R}_q(u_k) \leq \mathcal{R}_q(u)$. The coercivity of $\mathcal{R}_q$, once more, gives the boundedness of (u_k). A weakly convergent subsequence $u_k \rightharpoonup \bar{u}$ then satisfies $\|F(u_k) - v_k\|^p_{p,\varepsilon_k} \to 0$. $\varepsilon_k \to 0$ as well as $\delta_k \to 0$ imply $v_k \to v$, $\|F(u_k) - v\|^p_p \to 0$ and $F(\bar{u}) = v$ as well as $\mathcal{R}_q(\bar{u}) \leq \liminf_k \mathcal{R}_q(u_k)$. Hence $\bar{u}$ is also a $\mathcal{R}_q$-minimum solution and assumption 1 yields $\mathcal{R}_q(u_k - \bar{u}) = 0$. ■

DEFINITION 42 (*Generalized source condition*)

Let $F(u) = v$ obtain an $\mathcal{R}_q$-minimum solution $u^\dagger$ and assume that there exist $\beta_1, \beta_2, r, \sigma > 0$ and $\rho > \mathcal{R}_q(u^\dagger)$ such that

$$\mathcal{R}_q(u) - \mathcal{R}_q(u^\dagger) \geq \beta_1 \|u - u^\dagger\|_p^r - \beta_2 \|F(u) - F(u^\dagger)\|_p \tag{3.12}$$

for all $u \in \mathcal{D}(F)$ satisfying $\mathcal{R}_q(u) < \rho$ and $\|F(u) - F(u^\dagger)\|_p < \sigma$.

REMARK 43

In [28], Grasmair et al. show that 42 is indeed a generalization of convergence conditions in Banach spaces as described in [33, 12] and that it is equivalent to the standard source condition for linear and bounded forward operators.

If the generalized source conditions hold we can show convergence rates similar to the standard results in Banach spaces for linear and non linear operators. The source conditions do not include a tolerance and are unchanged from [28]. Using Proposition 48 we know that $\|F(u) - F(u^\dagger)\|_p \leq 2^{p-1}\left(\|F(u) - F(u^\dagger)\|_{p,\varepsilon}^p + \|\varepsilon\|_p^p\right)$. Hence, we assume $\varepsilon \leq 2^{1-p}\sigma$ and replace the condition $\|F(u) - F(u^\dagger)\|_p < \sigma$ by the stronger condition

$$\|F(u) - F(u^\dagger)\|_{p,\varepsilon} < (2^{1-p}\sigma - \|\varepsilon\|_p^p)^{1/p}.$$

This assumption also gives us an upper limit for ε to ensure that we can apply the source conditions.

LEMMA 44

Let $\|v^\delta - v\|_p \leq \delta$, $u^\delta \in \operatorname{argmin} J_{\alpha,\varepsilon}^\delta(u)$ and let Assumption 2 hold. For $\alpha \leq (2^{p-1}\beta_2)^{-1}$ we obtain the following estimates. For $p = 1$:

$$\|u^\delta - u^\dagger\|_p^r \leq \frac{(1 + \alpha\beta_2)\delta + \|\varepsilon\|_p}{\alpha\beta_1}$$
$$\|F(u^\delta) - v^\delta\|_p \leq \frac{(1 + \alpha\beta_2)\delta + \|\varepsilon\|_p}{1 - \alpha\beta_2}$$

For $p > 1$:

$$\|u^\delta - u^\dagger\|_p^r \leq \frac{\delta^p + \alpha\beta_2\delta + (\alpha\beta_2)^{\tilde{p}}/\tilde{p} + c\|\varepsilon\|_p^p}{\alpha\beta_1}$$
$$\|F(u^\delta) - v^\delta\|_p^p \leq \frac{\delta^p + \alpha\beta_2\delta + (\alpha\beta_2)^{\tilde{p}}/\tilde{p} + c\|\varepsilon\|_p^p}{1 - 1/p}$$

with $\frac{1}{p} + \frac{1}{\tilde{p}} = 1$.

Proof: We note that u^δ is a minimizer of $J^\delta_{\alpha,\varepsilon}$ and hence with $c = p\max(1, \|F(u^\delta) - v^\delta\|^{p-1}_{p,\varepsilon}, \|\varepsilon\|^{p-1}_p)$

$$\begin{aligned}\|F(u^\delta) - v^\delta\|^p_p + \alpha\mathcal{R}_q(u^\delta) &\leq \|F(u^\delta) - v^\delta\|^p_{p,\varepsilon} + \alpha\mathcal{R}_q(u^\delta) + c\|\varepsilon\|^p_p \\ &\leq \|F(u^\dagger) - v^\delta\|^p_{p,\varepsilon} + \alpha\mathcal{R}_q(u^\dagger) + c\|\varepsilon\|^p_p \\ &\leq \|F(u^\dagger) - v^\delta\|^p_p + \alpha\mathcal{R}_q(u^\dagger) + c\|\varepsilon\|^p_p \\ &\leq \delta^p + \alpha\mathcal{R}_q(u^\dagger) + c\|\varepsilon\|^p_p\ .\end{aligned}$$

Rearranging and inserting the source condition (Assumption 2) we obtain

$$\begin{aligned}\|F(u^\delta) - v^\delta\|^p_p + \alpha\beta_1\|u^\delta - u^\dagger\|^r - \alpha\beta_2\|F(u^\delta) - F(u^\dagger)\|_p \\ \leq \|F(u^\delta) - v^\delta\|^p_p + \alpha\left(\mathcal{R}_q(u^\delta) - \mathcal{R}_q(u^\dagger)\right) \\ \leq \delta^p + c\|\varepsilon\|^p_p.\end{aligned}$$

Inserting v^δ and using the triangle inequality yields

$$\begin{aligned}&\|F(u^\delta) - v^\delta\|^p_p + \alpha\beta_1\|u^\delta - u^\dagger\|^r \\ \leq&\delta^p + c\|\varepsilon\|^p_p + \alpha\beta_2\|F(u^\delta) - v^\delta\|_p + \alpha\beta_2\delta\end{aligned} \tag{3.13}$$

For $p = 1$ we have $c = 1$ and there holds

$$\alpha\beta_1\|u^\delta - u^\dagger\|^r_p \leq \delta(1 + \alpha\beta_2) + (\alpha\beta_2 - 1)\|F(u^\delta) - v^\delta\|_p + \|\varepsilon\|_p$$

which directly shows the assertion.

For $p > 1$ we apply Young's inequality for the terms $\beta_2\alpha$ and $\|F(u^\delta) - v^\delta\|_p$ to (3.13) resulting into

$$\left(1 - \frac{1}{p}\right)\|F(u^\delta) - v^\delta\|^p_p + \beta_1\alpha\|u^\delta - u^\dagger\|^r \leq \quad \delta^p + c\|\varepsilon\|^p_p + \beta_2\alpha\delta + \frac{1}{\tilde{p}}(\beta_2\alpha)^{\tilde{p}}$$

which completes the proof for $p > 1$. ■

3.5 Tolerances in L_p-Spaces

This section focuses on a particular case of Banach Spaces, the L_p-Spaces with $1 \leq p \leq 2$. These spaces are of particular interest in application since many applications are L_p-Spaces. Additionally, as used in SVR, tolerances can be directly adapted for these spaces. Therefore, it links the existing theory of support vector regression with the approach we propose for inverse problems in Hilbert spaces.

3.5.1 Definition and Properties

We recall the definition of ε-insensitive loss function (see, Vapniak, [72]) as used in SVR.

DEFINITION 45 (*ε-insensitive loss function*)

Let $1 \leq p < \infty$ and let $\varepsilon \in \mathbb{R}_0^+$. The ε-insensitive function is defined as

$$|\lambda|_\varepsilon := \begin{cases} 0, & |\lambda| \leq \varepsilon \\ |\lambda| - \varepsilon\,, & \textit{otherwise} \end{cases} \tag{3.14}$$

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

Subdifferentials The subdifferential for $|z|_\varepsilon$ can be calculated using the standard definition and looking at the three cases $z_0 = \varepsilon,\ z_0 = -\varepsilon,\ z_0 \neq |\varepsilon|$ For $z_0 \neq |\varepsilon|$ the function $|z|_\varepsilon$ is differential and we have

$$\partial|z|_\varepsilon = \begin{cases} -1, & z < -\varepsilon \\ 0, & -\varepsilon < z < \varepsilon\,. \\ 1, & z > \varepsilon \end{cases} \tag{3.15}$$

For the case $z_0 = \varepsilon$ all g that satisfy

$$|z|_\varepsilon - |z_0|_\varepsilon \geq g \cdot (z - z_0) \tag{3.16}$$

which is equivalent to

$$\max\{z - \varepsilon, 0\} - 0 \geq g \cdot (z - \varepsilon) \tag{3.17}$$

by the definition of $|\cdot|_\varepsilon$ and $z_0 = \varepsilon$. For $z < \varepsilon$ the inequality reduces to

$$0 \geq g \cdot \underbrace{(z - \varepsilon)}_{<0} \tag{3.18}$$

which is true for all $g \geq 0$. For $z \geq \varepsilon$ the inequality reduces to

$$z - \varepsilon \geq g \cdot \underbrace{(z - \varepsilon)}_{>0} \tag{3.19}$$

which holds true for $g \leq 1$. Therefore $g \in [0, 1]$ is the subdifferential at $z_0 = \varepsilon$. The case $z_0 = -\varepsilon$ is calculated analogously with the result $g \in [-1, 0]$ and therefore the subdifferential for $|\cdot|_\varepsilon$ is

$$\partial|z|_\varepsilon = \begin{cases} -1, & z < -\varepsilon \\ [-1, 0], & z = -\varepsilon \\ 0, & -\varepsilon < z < \varepsilon\,. \\ [0, 1], & z = \varepsilon \\ 1, & z > \varepsilon \end{cases} \tag{3.20}$$

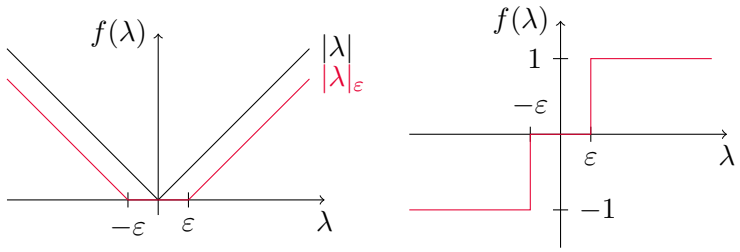

Figure 3.5: Subdifferential of $|z|_\varepsilon$

The figure 3.5 shows the ε-insensitive function for $\varepsilon = 0.4$ and the subdifferential. The ε-insensitive function can be extended for functions in $L_p(\Omega)$ via point-wise evaluation. The result is a so called (ε, L_p)-insensitive distance function.

DEFINITION 46 ((ε, L_p)-*insensitive distance function*)

Let $\Omega \subset \mathbb{R}^n$ and $1 \leq p < \infty$. For $\varepsilon \in \mathbb{R}_0^+$ the (ε, L_p)-insensitive distance function ($\|\cdot\|_{p,\varepsilon}^p$) is defined as

$$\|f\|_{L_p,\varepsilon}^p = \int_\Omega |f(x)|_\varepsilon^p \, \mathrm{d}x \tag{3.21}$$

for all $f \in L_p$.

REMARK 47 (*Properties*)

The ε-insensitive function fulfills all properties of a pre-distance metric. It is continuous, convex for $\varepsilon > 0$ and strict convex for $\varepsilon = 0$.

If $\varepsilon = 0$, it equals the traditional L_p-norm and fulfills the triangle inequality. For $\varepsilon > 0$, it does not fulfill the triangle inequality, and it cannot be a full metric.

Proof: Let $\varepsilon > 0$ and U a closed subset of Ω with non-zero finite measurements, i.e.

$$0 < \int_\Omega I_U(x)dx < \infty,$$

with I_U the index function

$$I_U(x) = \begin{cases} 1, & x \in U \\ 0, & \text{o.w.} \end{cases}.$$

Then for $f(x) := \frac{2}{3}\varepsilon I_U(x)$ we have

$$\|f\|_{L_p,\varepsilon}^p = 0$$

and

$$\begin{aligned}\|f+f\|_{L_p,\varepsilon}^p &= \int_\Omega \left|\frac{4}{3}I_U(x)\right|_\varepsilon^p dx \\ &= \int_\Omega \left|\frac{\varepsilon}{3}I_U(x)\right|^p dx \\ &\geq 0\end{aligned}$$

and thus

$$\|f+f\|_{L_p,\varepsilon}^p \geq \|f\|_{L_p,\varepsilon}^p + \|f\|_{L_p,\varepsilon}^p$$

which violates the triangle inequality. ■

REMARK 48

The ε-insensitive function fulfills the following inequalities which will be needed later for different proofs.

1. $\|f\|_{L_p,\varepsilon} \leq \|f\|_{L_P},\ \forall f \in \Omega$
2. $\|f\|_{L_p} \leq \|f\|_{L_p,\varepsilon} + \|\varepsilon\|_{L_p}$

The first inequality follows directly from the point-wise definition of the ε-insensitive function for functions in $L_p(\Omega)$ and the second inequality is derived by using the Minkowski inequality. The set Ω is divided into two disjunctive parts

$$\begin{aligned}\Omega_{f,\varepsilon} &:= \{x \in \Omega |\ |f(x)| > \varepsilon\} \\ \Omega_{f,0} &:= \{x \in \Omega |\ |f(x)| \leq \varepsilon\}\end{aligned}$$

where the ε-insensitive norm is zero on $\Omega_{f,0}$ and not zero on $\Omega_{f,\varepsilon}$. We assume without loss of generalization that none of the two sets is empty. If one of the sets is empty, the following proof reduces to one of the two cases depending on which set is empty.

For the set $\Omega_{f,\varepsilon}$ we have that

$$\begin{aligned}\|f\|_{L_p(\Omega_{f,\varepsilon})} &= \||f|\|_{L_p(\Omega_{f,\varepsilon})} \\ &= \||f| - \varepsilon + \varepsilon\|_{L_p(\Omega_{f,\varepsilon})} \\ &\leq \||f| - \varepsilon\|_{L_p(\Omega_{f,\varepsilon})} + \|\varepsilon\|_{L_p(\Omega_{f,\varepsilon})} \\ &= \|f\|_{L_p(\Omega_{f,\varepsilon}),\varepsilon} + \|\varepsilon\|_{L_p(\Omega_{f,\varepsilon})},\end{aligned}$$

by using the Minkowski inequality and the fact that $\||f| - \varepsilon\|_{L_p} = \|f\|_{L_p,\varepsilon}$ on $\Omega_{f,\varepsilon}$. For the set $\Omega_{f,0}$ we can not use the Minkowski inequality, instead we utilize the fact that

$$\|f\|_{L_p(\Omega_{f,0}),\varepsilon} = 0 \tag{3.22}$$

and

$$\|f\|_{L_p(\Omega_{f,0})} \leq \|\varepsilon\|_{L_p(\Omega_{f,0})} \tag{3.23}$$

on the set $\Omega_{f,0}$. Combining 3.22 and 3.23 results into

$$\begin{aligned}\|f\|_{L_p(\Omega_{f,0})} &\leq \|\varepsilon\|_{L_p(\Omega_{f,0})} \\ &= 0 + \|\varepsilon\|_{L_p(\Omega_{f,0})} \\ &= \|f\|_{L_p(\Omega_{f,0}),\varepsilon} + \|\varepsilon\|_{L_p(\Omega_{f,0})}.\end{aligned}$$

Since the sets $\Omega_{f,0}$ and $\Omega_{f,\varepsilon}$ are disjunctive and $\Omega = \Omega_{f,0} \cup \Omega_{f,\varepsilon}$, we have the second bound in remark 48.

In addition, the ε-insensitive function is weakly sequentially closed.

REMARK 49

The ε-insensitive function is w.l.s.c..

This statement follows directly from the two following lemmas as stated. The first lemma is about convex sets, and is needed when proving the second lemma.

LEMMA 50

Let $(X, \|\cdot\|)$ be a normed space with $M \subset X$ convex. Then M is weakly closed if and only if M is strongly closed.

Proof: $\Rightarrow$ Let M be weakly closed. Since X is a normed space, weakly closed and strongly closed are equivalent to weakly sequentially closed and strongly sequentially closed. Let $(x_k)_{k\in\mathbb{N}} \subset M$ be a strongly convergent sequence in X with limit point $x \in X$. Since every strongly convergent sequence is weakly convergent and M is weakly closed, i.e., the weak limit point of each weakly convergent sequence is an element of M, x is an element M as well. Thus M is strongly closed.

$\Leftarrow$ Let M be strongly closed. Assume that M is weakly open, then there exists a weakly convergent sequence $(y_n)_{n\in\mathbb{R}} \subset M$ with weak limit point $y \notin M$. Define $A = \{y\}$, which is a compact and convex set with $A \cap M = \{\emptyset\}$. Thus the Hahn-Banach separation theorem can be applied.

From the Hahn-Banach separation theorem we have that $\exists\varphi \in X^*$ and $t \in \mathbb{R}$ such that $\varphi(a) < t < \varphi(b)$, for all $a \in M$, $b \in A$. Therefore $\lim_{n\to\infty} \varphi(y_n) \neq \varphi(y)$, which is a contradiction to the weak convergence of y_n to y.

By contradiction, M has to be weakly closed as well. ■

LEMMA 51

Let V be a convex and compact set. Each convex and lower semicontinuous functional $f : V \to \mathbb{R}$ is weak lower semicontinuous.

Proof: We consider the epigraph which is defined as the set

$$\mathrm{Epi}(f) := \{(\lambda, u) \in \mathbb{R} \times \Omega |\, \lambda \geq f(u)\}.$$

Since f is convex and lower semicontinuous the epigraph is convex and strongly closed. To show the convexivity of $\mathrm{Epi}(f)$ we take $(\lambda_i, v_i) \in \mathrm{Epi}(f)$ for $i = 1, 2$, then from the convexity of f follows

$$\begin{aligned} t\lambda_1 + (1-t)\lambda_2 &\geq tf(v_1) + (1-t)f(v_2) \\ &\geq f(tv_1 + (1-t)v_2) \end{aligned}$$

and thus $(t\lambda_1 + (1-t)\lambda_2, tv_1 + (1-t)v_2) \in \mathrm{Epi}(f)$. Therefore $\mathrm{Epi}(f)$ is convex by the definition of a convex set.

Let $(\lambda_k, v_k)_{k\in\mathbb{N}} \subset$ be an arbitrary convergent sequence to the point $(\tilde{\lambda}, \tilde{v})$. This means that $\lambda_k \to \tilde{\lambda}$ and $v_k \to \tilde{v}$ for $k \to \infty$.

$$\begin{aligned} \tilde{\lambda} &= \lim_{k\to 0} \lambda_k \\ &= \liminf_{k\to 0} \lambda_k \\ &\geq \liminf_{k\to 0} f(v_k) \\ &\geq f(\tilde{v}), \end{aligned}$$

which means that the limit point $(\lambda_k, v_k) \in \mathrm{Epi}(f)$. Since the sequence was arbitrary the epigraph is closed.

From lemma 50 we have that the $\mathrm{Epi}(f)$ is convex and weakly closed. Let's assume now that f is not weakly semi-continuous, thus there exist a sequence $(u_k)_{k\in\mathbb{N}}$ weakly convergent to u and $\liminf_{k\to\infty} f(u_k) < f(u)$. Then there exists a subsequence $(u_{k_n})_{k_n\in\mathbb{N}}$ for which $\lim_{k_n\to\infty} f(u_{k_n}) = y \in \mathbb{R}$ and $y < f(u)$ hold. Since $\mathrm{Epi}(f)$ is weakly closed, the sequence $(f(u_{k_n}), u_{k_n}) \subset \mathrm{Epi}(f)$ converges weakly to $(y, u) \in \mathrm{Epi}(f)$ and $y \leq f(u)$, which is a contradiction. As a result f has to be weakly semi-continuous as well. ■

Altering the Tikhonov functional by utilizing the ε-insensitive function leads to the functional

$$J^{p,q}_{\delta,\alpha,\varepsilon}(u) = \frac{1}{p}\left\|F(u) - v^\delta\right\|^p_{L_p,\varepsilon} + \alpha\mathcal{R}_q(u). \tag{3.24}$$

3.6 Comparison to Ivanov- and Morozov-Regularization

Since it was introduced by Tikhonov, minimizing a cost functional, which consists of a discrepancy term and a regularization term, has been altered by many others. Morozov and Ivanov each introduced a minimization with non-linear constrains. Morozov's approach is to minimize the regularization penalty under constrain

$$\frac{1}{p}\left\|Fu - v^\delta\right\|^p_V \leq \tau\delta.$$

Ivanov's approach is to minimize the discrepancy term under constrain

$$\mathcal{R}_q(u) \leq \rho.$$

Both approaches minimize a functional and are summarized as

$$u_\tau^\delta = \underset{u \in \mathcal{D}(F)}{\operatorname{argmin}} \mathcal{R}_q(u), \quad \text{s.t.} \frac{1}{p} \left\| Fu - v^\delta \right\|_V^p \leq \tau\delta$$

with $\tau \geq 1$ fixed constant independent of δ for Morozov regularization and

$$u_\rho^\delta = \underset{u \in \mathcal{D}(F)}{\operatorname{argmin}} \frac{1}{p} \left\| Fu - v^\delta \right\|_V^p, \quad \text{s.t.} \mathcal{R}_q(u) \leq \rho$$

for Ivanov regularization.

These approaches seem similar, and in some cases, the reconstructions are the same for the best parameter choices. However, it can be shown that this statement is not valid in general. To demonstrate this, we will provide an example introduced to us by Kaltenbacher in 2015 at a summer school at the University of Bremen. It is based on an example given in [52]. While this example shows the difference between Morozov, Ivanov, and Tikhonov regularization, we will extend it to a more complex setting to illustrate the difference in our approach to incorporating tolerances.

For the example the operator F is defined as

$$\begin{aligned} F :& \mathbb{R} \to \mathbb{R} \\ & u \longmapsto (u-1)^3 \end{aligned}$$

and the true solution $u^\dagger = 1$. Thus the unnoised data is $v = 0$. We set the noised data as $v^\delta = 0.25$ and choose $R_q(u) = |u|$, which means $q = 1$. Additionally, $p = 2$ and $\varepsilon = 0.1$ are set. For Morozov and Ivanov exist parameters such that the solution of the regularization problem is at the true solution. For the classic Tikhonov method the Tikhonov functional has two local minima. The local mimima at the position less than 1.5 converges to the true solution at 1. However, for $\alpha = 0$ the Tikhonov functional does have only one minima that is not at the true solution. The global minimum does not converge to 1 bit 1.71 instead. At the true solution $u = 1$ the functional has a saddle point instead. For the Tikhonov regularization with discrepancy the global minimum of the the Tikhonov functional converges to the desired solution for $\alpha \to 0$.

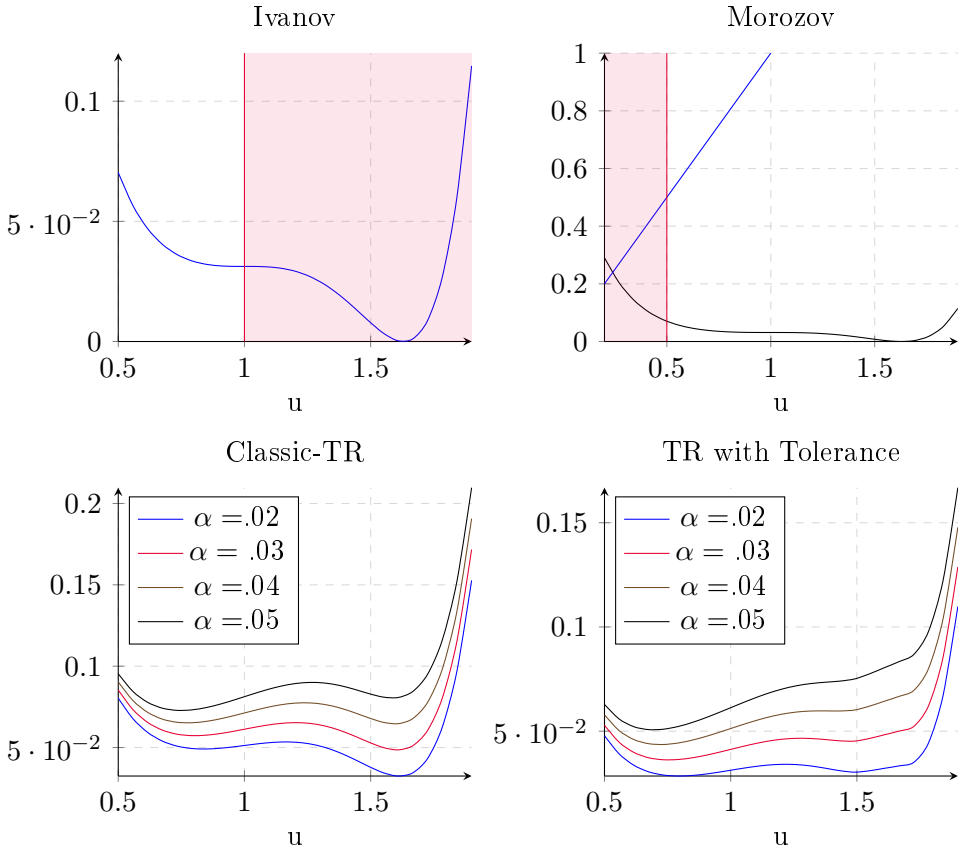

Figure 3.6: Example for different regularization methods.

3.7 Choosing ε

So far, we have introduced the concept of tolerances and their application for Tikhonov functionals and provided a mathematical framework. The following section will briefly outline how we can choose a tolerance threshold. We have two main ways to select a tolerance that we separate into two main groups: data-based a-priori information and an additional regularization parameter. If available, the first group includes methods based on statistical observations and multiple measurements. The second group integrates the tolerance to the regularization scheme.

We explained the difference between point estimations and interval estimates in the preliminaries. If we make a point estimation, such as the expected value when having multiple measurements, we ignore the additional available information from the observations. Instead, we can use the multiple measures to have an interval estimate of the measurement. One example is using the expected value and a confidence band. The ε-insensitive discrepancy term can model such an interval estimate by setting the expected value as the observed measurement v and the confidence band as $\varepsilon(t)$. The $\varepsilon(t)$ then depends on the method to calculate the confidence band as well as the confidence level.

The ε-insensitive distance regularizes the inversion problem by denoising the discrepancy term. This denoising is in the amplitude of the noise rather than the frequency. Low-, band- and high-pass filter denoise signals based on their frequency and therefore work differently than the with denoising ε-insensitive measures. We can use the parameter ε as an additional regularization parameter in the Tikhonov functional and choose it analogously to α.

We have shown that the introduced tolerance does, in fact, not reproduce the same result as the classical Tikhonov, Morozov, or Ivanov regularization. We consider a linear inverse problem to understand further, how the tolerance in discrepancy changes the reconstruction and thus yields better stability. The forward operator A maps from $\mathbb{R}^{500}$ to $\mathbb{R}^{500}$ and is a standard integral operator for the signal $u \in \mathbb{R}^{500}$. The ground truth signal is a composition of different cosine waves. For the noise model, we add two different noises. First, we add noise to the measurement using a Gaussian additive noise. Secondly, instead of applying the actual forward operator, we add a slight normal distributed noise on all non-zero coefficients of A. This noise on the operator is around 10^{-4} in magnitude.

We apply the adapted SVR-Solver for the numerical solution, which we will introduce in detail in the next chapter. For the parameters we choose $p = 1,\ q = 2,\ \alpha = 10^{-2},\ \varepsilon = \{0, 10^{-2}\}$. The solution for classical Tikhonov and altered Tikhonov with tolerance in discrepancy is shown in figure 3.7. The reconstruction in norm is 0.0525 for classical Tikhonov and 0.0202 when $\varepsilon = 10^{-2}$. From a visual inspection of the solution, we can see how the tolerance in discrepancy results in a step-wise approximation of the original signal.

The most significant error of the reconstruction is in the last points. This

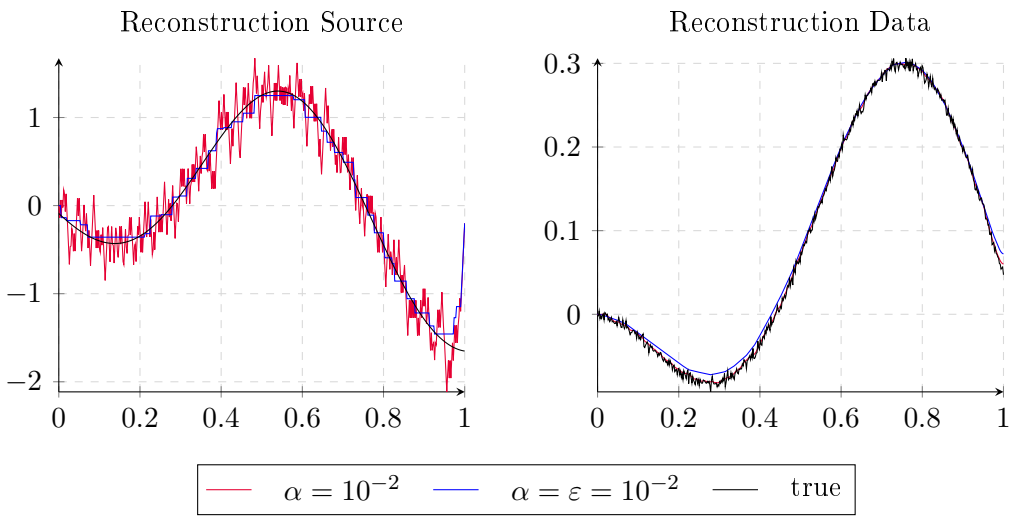

Figure 3.7: Example reconstruction .

behavior is expected when using an integral operator as the forward operator. For completion, we also compared the reconstruction error in the first 450 points, which is 0.0381 for Tikhonov and improves to 0.0042 when $\varepsilon = 10^{-2}$.

With this example ends chapter 3. In the next chapter we will have look at the numerical methods to find a minimizer of the altered Tikhonov functional.

4 Numerical Solution for L_p

This chapter gives a short overview of existing methods to numerically find a minimizer for the Tikhonov functional with tolerance in the discrepancy

$$J^{p,q}_{\delta,\alpha,\varepsilon}(u) = \frac{1}{p}\left\|F(u) - v^\delta\right\|^p_{L_p,\varepsilon} + \alpha\mathcal{R}_q(u) \tag{4.1}$$

with $1 \leq p \leq 2$ and $1 \leq q \leq 2$, and compares them with a new modified subgradient method derived in this thesis.

Due to the rising popularity of machine learning in recent decades, many numerical methods for training support vector machines have emerged [18]. These methods either work on the primal problem with linear kernels or the dual problem for different types of kernels. Additionally, methods for choosing kernels and hyperparameters have been developed [87].

Two aspects are important to consider when using numerical methods from machine learning. First, while the general idea is to minimize an error on a training set $S = \{x_i, y_i\}$ where often for support vector machines $x_i \in R^n$ and $y_i \in R^m$ or $y_i \in \{0,1\}^m$, the main goal is to have a stable solution that will make accurate predictions for new $x_j \notin S$. As a result, early stopping criteria are used to find approximate solutions which may not minimize the functional (4.1) but find a solution close to the desired minimization. This has the advantage that methods which with slow convergence to the exact minimum are still feasible if they converge fast to an approximation. Since overfitting on the training set S may lead to poor predictions on new data, these approximations are more stable with the right regularization, such as the error rate on a validation set. Secondly, one major challenge in machine learning, especially for image applications, is to process large amounts of sample data in the training set. In order to reduce training times and handle more training data, batch and mini-batch methods were developed, which are equivalent to coordinate or group coordinate descent in the dual for support vector machines.

For finding a minimizer of an ill-posed inverse problem it is crucial to overcome the problem's ill-posed nature and reconstruct the true underlying source. However, some machine learning-related methods, such as splitting training data into different sets for cross-validation, may not be feasible for

inverse problems if insufficient data is present. As a result, we are developing a new technique for solving the previously stated inverse problem with tolerances in discrepancy.

Of special interest are cases where $p \in \{1, 2\}$ as well as $q \in \{1, 2\}$. We present existing methods to solve inverse problems applicable to certain cases and other existing ones to solve missing cases such as for $p = q = 1$.

4.1 Solving with SVR-Solver

In the previous chapters we introduced support vector machines/regression and their solution. For support vector regression the functional

$$F(x_i) = \|f(x_i) - y_i\|_{\varepsilon,p}^p + \alpha\|f\|_2^2$$

has to be minimized for f, where $p = 1, 2$. For linear SVR $f(x_i) = \langle w, x_i\rangle + d$, or equally $f(x_i) = w^T x_i + d$ if $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. While this looks like a Tikhonov type minimizer, the difference is in the minimization parameter. For SVM / SVR, pairs of (x_i, y_i) are known, while in a inverse problem $f : X :\to Y$, y^δ is known. For linear discrete inverse problems, we can show that both formulations are in fact equal. Thus we can use algorithms for linear support vector regression to solve the inverse problem numerically.

In the linear discrete[1] inverse problem we have the forward model $f(x) = Ax$ with $A \in \mathbb{R}^{n\times d}$. By the definition of $\|\cdot\|_{\varepsilon,p}^p$ we have

$$\left\|Ax - y^\delta\right\|_{\varepsilon,p}^p = \sum_{i=1}^{d} \left\|x^T A_i^T - y_i^\delta\right\|_{\varepsilon,p}^p \tag{4.2}$$

where A_i denotes the i-th row of A. Using this equality we can minimize for x as if x is the hyperplane w for the SVR. Now the regularization is also applied on x instead of w. The new minimization functional is

$$F_{reg}(x) = \sum_{i=1}^{d} \left\|x^T A_i^T - y_i^\delta\right\|_{\varepsilon,p}^p + \alpha\left\|x\right\|^2 \tag{4.3}$$

and the training data for the SVR is the set of tuples (A_i^T, y_i^δ). This allows to directly apply existing methods of linear support vector regression to solve the inverse problem. This technique works for $p = 1, 2$, L_2-penalty, and linear forward operator.

In this thesis, the implementation uses python bindings of *libsvm* [16] and *liblinear* [24] as provided in the python module *SCIPY* [85].

While this method can be directly used for linear operators and L_2 regularization, we can not use it for other regularization terms.

[1] The discretization is in x, not y.

4.2 Non-Smooth Optimization

Solving a non-smooth optimization problem requires different numerical algorithms than the classical convex optimization problem. We can solve the SVM minimization problem by solving the dual problem as outlined in section 3.1.1. However, this approach is not feasible for $p = q = 1$ in (4.1) or other regularization terms such as TV regularization, as it does not yield a convex optimization problem.

The following section provides a short introduction to existing non-smooth solvers and their applications. Then, we use each algorithm to solve a numerical example of an ill-posed inverse problem to illustrate their capability for inverse problems.

4.2.1 Nelder-Mead Simplex Method

The Nelder-Mead simplex algorithm is an iterative algorithm to estimate a local extremum of a function by moving a simplex along the function surface. The simplex is changed by a heuristics on how to update the simplex vertices in each iteration. While being relatively stable and working on non-smooth functions, in practice, it is only feasible for dimensions up to 12 [30]. Gau et al. [25] show that adaptive parameters can improve the convergence and allow for larger input dimensions up to 40. In addition, theoretical results on convergence only exist for special cases and under strong assumptions [56]. This algorithm is one example of a group of algorithms known as direct search methods without gradient information [46]. Since they behave similarly in practice, we focus on this one method.

The basis for this algorithm is a simplex, defined as follows.

DEFINITION 52 (*Simplex*)

Given n linear independent vectors $x_0, x_1, \ldots, x_n \in \mathbb{R}^n$, then the convex hull

$$S := \left\{ \sum_{i=0}^{n} \lambda_i x_i \;\middle|\; \lambda_i \in \mathbb{R} \text{ und } \lambda_i > 0, \quad \sum_{i=0}^{n} \lambda_i = 1 \right\}$$

is called a n-dimensional simplex. The vectors $x_0, x_1, \ldots, x_n$ are referred to as simplex vertices.

In each iteration, the given simplex S is evaluated on the vertices, and the vertex with the highest value is moved along the line through itself and the center of the other simplex vertices in the input space. Three different cases are possible, and the best one is selected in each step. In the first case the new vertex lies on the same side as before and behind the original vertex. This case is called expansion. In the second case, the new vertex lies opposite of the original vertex. This case is called reflection. For case three, the new vertex lies within the simplex. This case is called contraction. Figure 4.1 illustrated the different cases on a simplex in a two-dimensional space.

Two stopping criteria are commonly used. Either the standard deviation of the function values along the simplex

$$\left(\frac{1}{n+1} \sum_{i=0}^{n} \left(f\left(x^{(k,i)}\right) - \bar{f}_k \right)^2 \right)^{1/2} < \tau_f,$$

falls below a given threshold $\tau_f > 0$. In the formula above

$$\bar{f}_k := \frac{1}{n+1} \sum_{j=0}^{n} f\left(x^{(k,j)}\right) \tag{4.4}$$

is the mean of the function values along the simplex vertices. The second criteria is the standard deviation of the simplex vertices in the input space

$$\left(\frac{1}{n+1} \sum_{i=0}^{n} \left\| x^{(k,i)} - \bar{x}_k \right\|^2 \right)^{1/2} < \tau_x,$$

falls below a given threshold $\tau_x > 0$. In the formula above

$$\bar{x}_k := \frac{1}{n+1} \sum_{j=0}^{n} x^{(k,j)} \tag{4.5}$$

is the geometric mean of the simplex vertices. Both criteria can be used in combination as well. We summarize the full algorithm below.

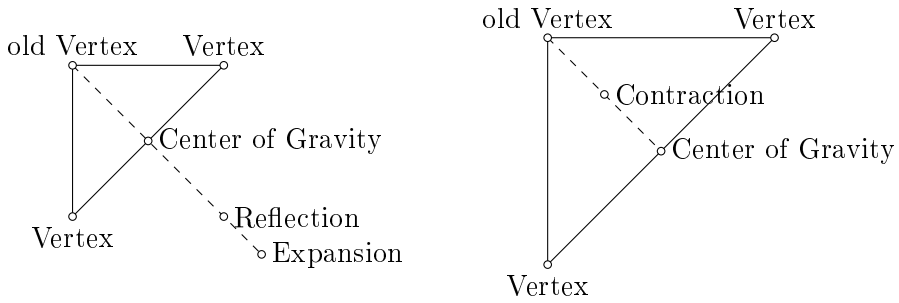

Figure 4.1: Example for reflection, expansion, and contraction of a simplex.

Algorithm 1: Nelder Mead

Data: f, $0 < \alpha < 1, \beta > 1$, $0 < \gamma \leq 1$ and $x^{(0,0)} \in \mathbb{R}^n$
Result: x minimum of f
$k = 0$ and generate S_0;
for $k \leq k_{max}$ *and* $\left(\frac{1}{n+1}\sum_{i=0}^{n}\left(f(x^{(k,i)}) - \bar{f}_k\right)^2\right)^{1/2} < \tau_f$ **do**
  $f(x^{(k,m)}) = \max\{f(x^{(k,i)}) | \, i = 0, \ldots, n\}$;
  $f(x^{(k,l)}) = \min\{f(x^{(k,i)}) | \, i = 0, \ldots, n\}$;
  $s^{(k,m)} = \frac{1}{n}\sum_{i=0, i\neq m}^{n} x^{(k,i)}$;
  reflect to find $x^r = s^{(k,m)} + \gamma(s^{(k,m)} - x^{(k,m)})$;
  if $f(x^r) < f(x^{(k,l)})$ **then**
    perform expansion $x^e = s^{(k,m)} + \beta\left(x^r - s^{(k,m)}\right)$;
    set $x^{(k+1,m)} = \min\{f(x^r), f(x^e)\}$
  end
  if $f(x^{(k,l)}) \leq f(x^r) \leq \max\{f(x^{(k,i)}) | \, i = 0, \ldots, n, \, i \neq m\}$ **then**
    set $x^{(k+1,m)} = x^r$
  end
  if $f(x^r) > f\left(x^{(k,m)}\right)$ **then**
    perform partial contraction $x^c = s^{(k,m)} + \alpha\left(x^{(k,m)} - s^{(k,m)}\right)$;
    if $f(x^c) < f\left(x^{(k,m)}\right)$ **then**
      $x^{(k+1,m)} = x^c$
    end
    else if $f(x^c) \geq f\left(x^{(k,m)}\right)$ **then**
      perform total contraction $x^{(k+1,i)} = \frac{1}{2}(x^{(k,i)} + x^{(k,l)})$, $i \neq l$
    end
    else
      $x^r = x^{(k+1,m)}$
    end
  end
end

4.2.2 Genetic Algorithms

Genetic algorithms (GAs) are a class of optimization techniques used to solve NP-hard problems, such as the traveling salesman problem. They were first introduced by John H. Holland and are often applied in discrete optimization problems and can be used for continuous problems as well. Genetic algorithms are used to train topologies of neural networks. Example algorithms are the NEAT [75], and HYPER-NEAT [74] algorithms.

Genetic algorithms do not rely on continuity or convexity of the minimization functions. Therefore they can be used to solve non-smooth, non-continuous, and discrete functions. They are well suited for optimization problems with many local but few global optima. Furthermore, genetic algorithms can be used for problems with large input dimensions (see, for example, [26]). Their convergence, however, is slow, and the algorithm needs a large number of function evaluations. Studies on the convergence of genetic algorithms include [22], where a general framework based on Markov chains is introduced and global convergence shown.

We must decode the input x in a so-called genome to minimize a function $f(x)$ with a genetic algorithm. As an example, this genome can be a permutation of the stops for the traveling salesman problem. For a continuous problem, the gene equals to the input $x \in \mathbb{R}^n$. The algorithm starts with a set of inputs x, called a population. In each iteration, we sort this population by their fitness. For minimization, we sort the values in ascending order by function value. The best ones are used to generate a new population via mating and mutation. The main parameters for this algorithm are the size of population s_p, number of iterations k_{max}, mutation rate m_r and mutation range m_s. We can apply different strategies for crossover and mutation and optional parameters as well. For this work, we focus on continuous problems and use the following strategy for selection, crossover, and mutation.

Crossover Each gene has the same length $n \in \mathbb{N}_+$. For crossover between the pair $(x_i,\ x_j)$ a random vector r sampled from $U([0,\ 1]^n)$ is generated and the new gene is calculated through

$$x_{new,1} = x_i + r(x_j - x_i)^T.$$

Mutation Genes are mutated randomly based on the mutation rate and mutation range. For a gene $x \in R$ the mutation chance is given by α sampled from $U([0,\ 1]^n)$. The mutation is then applied on the subset of genes $S = \{x_k \mid \alpha_k >= m_r\}$ that has a mutation chance higher than the given threshold m_r. The mutated gene is calculated via

$$\hat{x_k} = x_k + \frac{m_s}{2}\beta,\ \beta \sim U([0,1]),\ \text{for all } x_k \in S.$$

Algorithm 2: Genetic Optimization

Data: f, m_r, s_p, k_{max}
Result: x minimum of f
initialize population random;
store current best $f_{best} = f(x_0)$;
$k = 0$;
for $k \leq k_{max}$ **do**
    sort population by fitness;
    keep only best n-genes;
    cross genes for new population;
    mutate genes;
end

4.2.3 Particle Swarm Optimization

Based on the behavior of a bird flock, the original particle swarm optimization (PSO) uses a swarm of particles that share information among themselves to find a (local) minimum iteratively. Many adaptations [8, 59, 88] have been done over the years, but the core principle stayed the same. Each particle has a position $x_{p,k} \in \mathbb{R}^n$ a velocity $v_{p,k} \in \mathbb{R}^n$ and best visited point $x_{p,best} \in \mathbb{R}^n$. The minimize function f (also often referred to as energy function) evaluates current positions. A swarm consists of a fixed number $m \in \mathbb{N}_+$ of these particles and a memory of the best-found solution of the whole swarm $x_{best} \in \mathbb{R}^n$. Based on a weighted sum of velocity, distance to best own position, best swarm position and swarm center, a new velocity for the particle is calculated in each iteration. Using this velocity all positions and the best known positions are updated. In each iteration the particles move according to

$$\begin{aligned} g_{p,k} &= x_{p,best} - x_{p,k} \\ q_{p,k} &= x_{best} - x_{p,k} \\ r_{p,k} &= \left(\frac{1}{m}\sum x_p\right) - x_{p,k} \\ v_{p,k+1} &= s_1 v_{p,k} + s_2 g_{p,k} + s_3 q_{p,k} + s_4 r_{p,k} \\ x_{p,k+1} &= x_{p,k} + s v_{p,k+1} \end{aligned}$$

and the best solution found for the swarm will be updated. This method was designed to train neural networks originally [41], as it is less effected by local saddle points and local minima compared to classical gradient methods. PSO algorithms are used for graph optimization and power grid network layouts until today [15, 92].

Yang et al. [89] use an accelerated PSO algorithm to solve nonlinear SVMs for business problems. A two-step iteration scheme is applied where PSO estimates optimal kernel parameters, and then the SVM is calculated using the

PSO algorithm again. These two iterations steps are repeated until convergence. In [48], a survey about PSO algorithms for feature selection, including feature selection for SVMs, is performed.

Algorithm 3: Particle Swarm Optimization

Data: f, number particles, number of maximal iterations i_{max}
Result: x minimum of f
initialize swarm $\{p_i\}$ with random position and velocity;
store current best $f_{best} = \min\{f(p_i)\}$;
$i = 0$;
for $i < i_{max}$ **do**
  distance to personal best;
  $g_{p,i} = x_{p,best} - x_{p,i}$;
  distance to swarm best;
  $q_{p,i} = x_{best} - x_{p,i}$;
  distance to swarm centroid;
  $r_{p,i} = \left(\frac{1}{m}\sum x_p\right) - x_{p,i}$;
  calculate velocity for each particle;
  $v_{p,i+1} = s_1 v_{p,i} + s_2 g_{p,i} + s_3 q_{p,i} + s_4 r_{p,i}$;
  move particles;
  $x_{p,i+1} = x_{p,i} + s v_{p,i+1}$;
  $f_{best} = \min\{f_{best}, f(p_0), \cdots, f(p_i)\}$;
  $i++$;
end

4.2.4 Simulated Annealing

Simulated annealing is a method for approximating a global solution to an optimization problem. It is used to solve problems with many local extrema and is an iterative method. The method was described independently by Kirkpatrick et al. (1983) and by Cerny (1985) [66] and can be interpreted as an improved hill-climbing method. Simulated annealing allows steps with less optimal solutions to avoid local extrema. In the beginning, such a decrease in the optimality condition can be large, but the acceptance rate will drop over each iteration, and the method converges. A decrease in optimality will be an increase in the function value at the next iteration if the optimality criteria is finding a minimum, i.e. $f(x_{k+1}) > f(x)$ when a minimum of $f(x)$ is desired. Since we consider minimization problems in our work, we will only consider this case as an optimality condition.

In each iteration, k, a random neighbor $k_c \in B(x_{x_k})$ is selected and accepted if $f(x_c) \leq f(x_k)$ and $x_{k+1} = x_c$ updated. If $f(x_c) > f(x_k)$, the new point is accepted with a probability of

$$\exp\left(-\frac{f(x_c) - f(x_k)}{T_t}\right)$$

where $T_t \in \mathbb{R}_+$. The value $T_t \in \mathbb{R}_+$ is referred to as the temperature of the system and reduced in each iteration until it converges to zero. At this point the algorithm terminates and the best found solution is returned as an approximation of the global minimum.

The name simulated annealing is based on the decrease of acceptance rate for less optimal solutions, it is often described as cooling down a system to reduce jumpy behavior. The method is frequently used with other optimization approaches such as random walk and Markov chains to compute random neighbors in each iteration.

Simulated annealing is used for SVM algorithms to identify good kernel functions and parameters [91, 51, 67]. In [66], simulated annealing is directly used to solve a least squares twin support vector machine (LSTSVM).

In order to improve convergence speed and reduce the number of needed function evaluations, we altered the classical formulation by shrinking the neighborhood area in each iteration. This alteration reduces the number of tries needed to find an acceptable solution and forces the algorithm to converge faster to a local minimum.

Algorithm 4: Simulated Annealing Algorithm

Data: f, x_0, T_0, c
Result: x minimum of f
set temperature $T = T_0$;
store current best $f_{best} = f(x_0)$;
$k = 0$;
while $T > 0$ **do**
  x_c = random neighbor of x_k;
  if $f(x_k) > f(x_c)$ **then**
    $x_{k+1} = x_c$;
  else
    set $x_{k+1} = x_c$ with probability $\exp\left(-\frac{f(x_c)-f(x_k)}{T_t}\right)$;
  end
  $T = T * c$;
  $f_{best} = \min\{f_{best}, f(x_{k+1})\}$;
  $k++$
end

4.2.5 Summary

We briefly introduced a range of non-linear and non-smooth numerical solvers. While many of them have been applied to solve SVMs directly or support other methods to find a suitable solution, none of the above methods are feasible for solving large-scale inverse problems. This is due to their slow convergence for large-scale problems and the absence of suitable convergence criteria that define the termination of the iterations.We provide example applications on

toy examples at the end of this chapter, where we compare the different numerical methods to each other. As a result, in theory, the above methods are able to solve the inverse problem but are too slow or inaccurate in practice to be feasible.

For large scale machine learning problems, other methods based on stochastic gradients are used. We will have a look at this type of optimization problem in the following section.

4.3 Stochastic Gradient Descent

Stochastic gradient methods are an essential tool in training weights of neural networks. Neural networks are trained with backpropagation, a chain rule-based (sub-)gradient calculation method. Sutskever et al. showed in [76] that using stochastic gradients and momentum do improve the learning rates of neural networks.

Well known stochastic gradient descent methods are Adaptive Gradient Algorithm (AdaGrad) [20], Root Mean Square Propagation (RMSProp) [62] and Nesterov Accelerated Momentum [9]. All these methods are first-order methods, meaning they only use the first-order derivative in each iteration step. They do, however, include second-order information indirectly via momentum. We take the ADAM optimizer to represent stochastic gradient methods with adaptive step size and momentum for a closer look and comparison.

Stochastic gradients are defined on functions f that are a sum of other functions f_m. The definition is given below.

DEFINITION 53 (*Stochastic Gradient*)

If the function $f(x)$ can be written as

$$f(x) = \sum_{m}^{M} f_m(x)$$

for all $x \in \mathcal{D}(f)$ and $i > 1$, then the stochastic gradient is a random sample of the set $\{\nabla f_1(x), \ldots, \nabla f_M(x)\}$. A subset Θ of unique stochastic gradients is called a mini batch. The (batched) stochastic gradient of f at x is denoted with $\nabla_\Theta f(x)$.

We can use stochastic gradients for minimization by applying an iterative scheme

$$x_{k+1} = x_k - \frac{\eta}{M} \nabla_{\Theta_k} f_{m_k}(x)$$

where $\eta > 0$ and $f_{m_k} \in \{f_1, \ldots, f_M\}$ is a random sample uniformly drawn. In practice the set $\{f_1, \ldots, f_M\}$ is shuffled and then each f_m is selected in order. The set is reshuffled when the end is reached. This iterative method is called stochastic gradient descent. We can use this method with mini-batches as

well. Stochastic gradient descent has been proven to converge almost surely in [45, 78].

Stochastic gradient descent methods are not descending in general, and they are an unbiased estimate of the entire gradient. They are often used in machine learning where large quantities of training data have to be processed. The stochastic gradient is calculated on a subset of the given training data for machine learning. This way, large datasets can be processed in parallel and without memory issues.

While stochastic gradient descent is faster to calculate for one iteration, they have slower convergence than complete gradient methods and need a-priori step-size choice for η. Additionally, if a stochastic gradient $\nabla_\Theta f_m(x)$ is zero, this does not indicate that the entire gradient is zero and x is a (local) extrema.

Different adaptions to the standard stochastic gradient method have been proposed that speed up convergence. Kingma and Ba introduce one stochastic gradient method called ADAM in their paper [43]. We will look at ADAM since it incorporates two ideas for faster convergence and is commonly used to train deep neural networks. Firstly, ADAM uses momentum in the (stochastic) gradients. For the momentum the iterative step is taken into a weighted sum of current gradient and former gradients. The iteration step becomes

$$x_{k+1} = x_k - \eta m_k$$

where

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1)\nabla_{\Theta_k} f(x)$$

Kingma and Ba refer to this as the first momentum. Secondly, the ADAM uses an exponential moving average on the gradients, similar to RMSProp[2]. For the exponential moving average the update rule is

$$x_{k+1} = x_k - \frac{\eta}{\sqrt{v_k + \varepsilon}} \odot \nabla_{\Theta_k} f(x)$$

with

$$v_k = \beta_2 v_{k-1} + (1 - \beta_2)\nabla_{\Theta_k} f(x) \odot \nabla_{\Theta_k} f(x),$$

where $\varepsilon > 0$ is a fixed constant to avoid division by zero. In the above expressions, $\odot$ and $\oslash$ denote the Hadamard product and Hadamard division, respectively. ADAM combines the first and second momentum vector in an unbiased way. First m_0, s_0 are initialized with zeros elements. Then the

[2]RMSProp is an adaptive learning rate optimizer proposed by Geoff Hinton in his lecture *Neural Networks for Machine Learning*. It is unpublished but used for training neural networks and used in papers such as [42, 65]. Further analysis of RMSProp can be found in [17].

iteration update is

$$\begin{aligned}
\beta_{1,k} &= \beta_1 \lambda^k \\
m_{k+1} &= \beta_{1,k} m_k - (1 - \beta_{1,k}) \nabla_{\Theta_k} f(x_k) \\
s_{k+1} &= \beta_2 s_k + (1 - \beta_2) \nabla_{\Theta_k} f(x_k) \odot \nabla_{\Theta_k} f(x_k) \\
\hat{m} &= \frac{m_{k+1}}{1 - \beta_1^t} \\
\hat{s} &= \frac{s_{k+1}}{1 - \beta_2^t} \\
x_{k+1} &= x_k + \eta \hat{m} \oslash \sqrt{\hat{s} + \varepsilon}.
\end{aligned}$$

This method works on stochastic gradients as well as complete gradients. Kingma and Ba suggest $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$ as good default settings. It is important to note that the default settings are for machine learning with deep neural nets where inputs are scaled, and normalization layers are used and may not be a good choice for non-normalized objective functions.

Stochastic gradient methods are more suited for large-scale inverse problems since they are less affected by input dimensions. However, they depend on an appropriate parameter choice and lack a reliable convergence criterion, except the complete gradient is calculated in each iteration. The following section will examine subgradients which are a generalization of gradients. The stochastic gradient method can be applied to subgradients and can then be used to minimize non-smooth objective functions.

Algorithm 5: ADAM Optimizer

Data: f, β_1, β_2, η, k_{max}
Result: x argmin of f
initialize $m_0 = 0$ and $s_0 = 0$;
$k = 0$;
while $k < k_{max}$ **do**
  calculate stochastic gradient $\nabla_{\Theta_k} f(x_k)$ $\beta_{1,k} = \beta_1 \lambda^k$;
  $m_{k+1} = \beta_{1,k} m_k - (1 - \beta_{1,k}) \nabla_{\Theta_k} f(x_k)$;
  $s_{k+1} = \beta_2 s_k + (1 - \beta_2) \nabla_{\Theta_k} f(x_k) \odot \nabla_{\Theta_k} f(x_k)$;
  $\hat{m} = \frac{m_{k+1}}{1 - \beta_1^t}$;
  $\hat{s} = \frac{s_{k+1}}{1 - \beta_2^t}$;
  $x_{k+1} = x_k + \eta \hat{m} \oslash \sqrt{\hat{s} + \varepsilon}$;
end

4.4 Subdifferential and Subgradients

Subgradients generalize gradients for convex functions. We defined them in the preliminaries and provided some additional information. For a short summary, they are defined as stated below.

DEFINITION 54 (*Subgradient and Subdifferential*)

Let M be a convex subset of a real Banach space X and $f : M \to \mathbb{R}$ a convex functional. An element $x^* \in X^*$ is a subgradient of f in x_0 if

$$f(x) \geq f(x_0) + \langle x^*, x - x_0 \rangle \quad \forall x \in M$$

and $f(x_0) \neq \pm\infty$. The subdifferential of f at x_0 is denoted as $\partial f(x_0)$ defined as

$$\partial f(x_0) := \{x^* \in X^* | \, f(x) \geq f(x_0) + \langle x^*, x - x_0 \rangle \quad \forall x \in M\}$$

which is the set of all subgradients of f in x_0.

In contrast to gradient methods, subgradient methods are not descent methods. Consider the iterative update

$$x_{k+1} = x_k + s_k d$$

with s_k as the step size and d as the step direction. For a descent method there exists $c > 0$ such that the inequality $f(x_k) < f(x_{x+1})$ holds for all $s_k < c$. For subgradient methods we choose $d \in \partial f(x_k)$. However if d is a subgradient of f at x_k we cannot ensure that there exists $s_k > 0$ such that $f(x_{k+1}) < f(x_k)$ and thus the sequence $f(x_k)_{k \in \mathbb{N}}$ is not monotone decreasing. We summarize this in a proposition and prove our statement.

PROPOSITION 55

Subgradients are not a descent direction and thus the following is not true. For each $x \in \mathcal{D}(f)$ in $\exists \varepsilon > 0$ s.t. $\forall 0 < s < \varepsilon$:

$$f(x) \geq f(x + h \cdot g), \; g \in \partial f(x).$$

Proof: We use the definition of subgradients to show this claim. Let $g \in \partial f(x)$ and $s > 0$ then from the definition of subgradients we have

$$f(x - s \cdot g) \geq f(x) + \langle g, -s \cdot g \rangle$$

and it follows that $f(x - s \cdot g)$ is only bounded below not above. To show that we can not find an upper bound we take a counter example.

This is best shown by contradiction using an example. The example can be found with different values in [10]. Let $x = (y, z) \in \mathbb{R}^2$ and $f(x) = f(y, z) = |y| + 3|z|$, then at $x_0 = (1, 0)$, $g = (1, 3) \in \partial f(x_0)$. Since $s > 0$ we have

$$\begin{aligned} f(x_0 - sg) &= |1 - s| + 3|s| = |1 - s| + 3s \\ &\geq |1 - s + 3s| = |1 + 2s| = 1 + 2s \\ &> 1 = f(x_0) \end{aligned}$$

and thus $-g$ is not a descending direction. ■

In contrast to gradients we can not derive an upper bound to ensure a descent in function value for small enough step sizes. This is an important difference when applying subgradients in numerical minimization.

Similar to gradients, subgradients can be used in an iterative scheme to minimize a convex function. To minimze a convex function f, starting from an initial guess $x_0 \in \mathcal{D}(f)$, the iterations steps are

$$x_{k+1} = x_k - s_k \cdot g_k, \tag{4.6}$$

where $g_k \in \partial f(x_k)$ and $s_k > 0$ is the step size. The iteration stops if one of the following three criteria is fulfilled. First, $0 \in \partial f(x_k)$. Second, the function value $f(x_k)$ is less then a defined tolerance. Third, a maximum number of iterations is exceeded. Contrary to gradient descent, the subgradient method is not a descent method. Therefore the step size needs to be defined a-priori and we keep track of the best solution in each iteration

$$f_{best}^{(k+1)} = \min\{f_{best}^{(k)}, f(x_{k+1})\}.$$

Four classical choices are most commonly used for step size rules. These are

1. Constant step size: $s_k = c$ is constant for all k
2. Constant step length: $s_k = c/\|g_k\|_2$
3. Square summable but not summable:

   $$\sum_{k=0}^{\infty} s_k^2 < \infty, \text{ and } \sum_{k=0}^{\infty} s_k = \infty.$$

   One common choice for this step size rule is $s_k = a/(b+k)$, where $a > 0$ and $b \geq 0$.
4. Nonsummable diminishing:

   $$\lim_{k\to\infty} s_k = 0, \text{ and } \sum_{k=0}^{\infty} s_k = \infty.$$

   This rule is also called *diminishing step-size rule* and a typical example is $s_n = a/\sqrt{k}$, where $a > 0$.

Algorithm 6: Classic subgradient method

Data: $\partial f, x_0, (s_n)_{n\in N}$
Result: u minimum of f
$k = 0$;
$f_{best} = f(x_0)$;
while $0 \notin \partial f(x)$ **do**
  $x_{k+1} = x_k - s_k \cdot g_k, \quad g_k \in \partial f(x_k)$;
  $f_{best} = \min\{f_{best}, f(x_{k+1})\}$;
  $k++$
end

4.4.1 Adaptive Stepsize

To overcome the restrictions of a fixed a-priori step size we determine s_{k+1} via

$$s_{k+1} = \frac{s_0}{\alpha_k} \tag{4.7}$$

and for α_{k+1}

$$\alpha_{k+1} := \begin{cases} \alpha_k & f(x_{k+1}) < f(x_k) \\ \alpha_k + 1 & \text{otherwise} \end{cases}. \tag{4.8}$$

The iteration is terminated based on three criteria:

- the subgradient is zero
- the decrease in function value, $f(x_k) - f(x)$, falls below a given threshold
- the step size, $\|s_k g_k\|$, diminishes, i.e. $\|s_k g_k\| < tol$, where $tol > 0$

Only the first condition ensures a local minimum is found while the second and third condition ensure numerical stability and determination of the algorithm.

Algorithm 7: Subgradient with adaptive decreasing step size

Data: $f, \partial f, x_0, s_0, k_{max}$
Result: x minimum of f
$k = 0$;
$f_{min} = f(x_0)$;
$x_{min} = x_0$;
while $0 \notin \partial f(x)$ *and* $k < k_{max}$ **do**
  $x_{temp} = x_k - \frac{s_0}{\alpha_k} \partial f(x_k)$;
  if $f(x_{temp}) > f(x_k)$ **then**
    $\alpha_{k+1} = \alpha_k + 1$;
    $x_{k+1} = x_k - \frac{s_0}{\alpha_{k+1}} \partial f(x_k)$;
  else
    $\alpha_{k+1} = \alpha_k$;
    $x_{k+1} = x_{temp}$
  end
  if $f(x_{k+1}) < f(x_{min})$ **then**
    $x_{min} = x_{k+1}$;
    $f_{min} = f(x_{min})$;
  end
  $k++$
end

We changed the step-size rule from an a-priori choice to an adaptive method during the minimization iterations. Therefore, the convergence is not covered by existing convergence proofs, which analyze the four different step-size rules separately. In the following proposition, we summarize our convergence criterium and prove it.

PROPOSITION 56

Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex, and has a minimizer $\tilde{x}$. Assume that the subgradients are bounded in norm, $\|g_k\|_2 < c$, $c > 0$, for all k.

i) For a bounded step size s_k there exists an index $\beta \in \mathbb{N}$ such that the iteration scheme in (4.6), (4.7), and (4.8) converges at least to the minimizer within $\frac{c^2\beta}{2}$ with $\beta \leq s_0$.

ii) Let the step size s_k satisfy $\lim_{k\to\infty} s_k = 0$. Then the iteration scheme in (4.6), (4.7), and (4.8) converges.

Proof: For the proof we follow [90] and add alterations for the adaptive step-size rule when necessary. Let $\tilde{x}$ be an arbitrary optimal point, then from the definition of subgradients we have

$$\begin{aligned}\|x_{k+1} - \tilde{x}\|_2^2 &= \|x_k - s_k g_k - \tilde{x}\|_2^2 \\ &= \|x_k - \tilde{x}\|_2^2 - 2s_k < g_k, x_k - \tilde{x} > + s_k^2 \|g_k\|_2^2 \\ &\leq \|x_k - \tilde{x}\|_2^2 - 2s_k(f(x_k) - f(\tilde{x})) + s_k^2\|g_k\|_2^2\end{aligned}$$

which leads to the following inequality by recursion

$$\|x_{k+1} - \tilde{x}\|_2^2 \leq \|x_1 - \tilde{x}\|_2^2 - 2\sum_{i=1}^{k} s_i(f(x_i) - f(\tilde{x})) + \sum_{i=1}^{k} s_i^2\|g_i\|_2^2.$$

We can drop the left side since it is positive and conclude

$$0 \leq \|x_1 - \tilde{x}\|_2^2 - 2\sum_{i=1}^{k} s_i(f(x_i) - f(\tilde{x})) + \sum_{i=1}^{k} s_i^2\|g_i\|_2^2. \tag{4.9}$$

Considering that the minimum of the set $\{f(x_i) - f(\tilde{x})\}$ is in fact

$$\min_{i=0,\cdots,k} f(x_i) - f(\tilde{x}) = \left(\min_{i=0,\cdots,k} f(x_i)\right) - f(\tilde{x})$$

and rearranging equation (4.9) we obtain the following bound

$$\min_{i=0,\cdots,k}\{f(x_i)\} - f(\tilde{x}) \leq \frac{\|x_0 - \tilde{x}\|_2^2 + \sum_{i=0}^{k} s_i^2\|g_i\|_2^2}{2\sum_{i=0}^{k} s_i} \tag{4.10}$$

The classical proof uses a predefined step size rule such as constant step size or diminishing step size to show convergence. For our approach, we need to consider two cases. Either the sequence $(s_k)_{k\in\mathbb{N}}$ does not converge to zero or or it does. In both cases, the sequence $(s_k)_{k\in\mathbb{N}}$ is monotonically decreasing and strictly positive.

For the first case, we assume that the sequence $(s_k)_{k\in\mathbb{N}}$ does not converge to zero. We know that the sequence is monotonically decreasing, and strictly positive. Thus it is bounded below by a positive number $\phi > 0$. Since the

sequence is constructed from a fixed ruleset, there exists an index β for which $s_k = s_\beta$ with $k \geq \beta$. Thus the step size rule becomes a fixed step size and converges to an optimal solution within $\frac{c^2 s_\beta}{2}$, i.e.

$$\left(\min_{i=0,\cdots,k} f(x_i)\right) - f(\tilde{x}) \leq \frac{c^2 s_\beta}{2}, \tag{4.11}$$

which proves the (i) in proposition 56.

For proving (ii), namely, that $(s_k)_{k\in\mathbb{N}}$ converges to zero, we can find a monotone subsequence $(s_j)_{j\in\mathbb{N}}$ such that $s_{j+1} < s_j$. By the construction rule of the sequence we know that it is nonsummable diminishing. Since $(s_k)_{k\in\mathbb{N}}$ converges to zero and $s_k \geq s_j$ the original sequence is nonsummable diminishing as well. Thus a classical criteria to ensure that (4.10) converges to zero is ensured. It follows that $\min_{i=0,\cdots,k} f(x_i) \to f(\tilde{x})$ for $k \to \infty$. ■

From the proof, we see that by adding theconstraint

$$\|s_k g_k\|_2^2 < c_1,\ c_1 > 0$$

to (4.6) leads to a maximum change in x in each iteration even if $\|g_k\|_2^2$ is unbounded. We will discuss the effect of such an additional bound on converge and numerical stability in future publications. We restrict the proof to convex functions to ensure the existence of a subgradient in each iteration. However, a more general approach combining subgradients and regular gradients is possible for nonconvex functions. The necessary subgradients for our method are either calculated analytically or can be approximated numerically.

4.4.2 Numerical Investigation

We developed the adaptive step-size method to reduce the importance of the initial step-size and step-size rule as well as to improve convergence speed of the subgradient iterative minimizer. This section investigates if the adaptive step-size has an advantage over the regular subgradient method and ADAM. We choose the function

$$f(x) = \begin{cases} |x| & |x| \leq 1 \\ 2|x| - 1 & \text{otherwise} \end{cases} \tag{4.12}$$

since it is convex, continuous everywhere but not differential at $x \in \{-1, 0, 1\}$. The global minimum is at $x = 0$ and this point is also the only local minimum. The subdifferential is given as

$$\partial f(x) = \begin{cases} 2 \cdot \operatorname{sgn}(x) & |x| > 1 \\ [2, 1] & x = 1 \\ [-1, -2] & x = -1 \\ \operatorname{sgn}(x) & \text{otherwise} \end{cases}. \tag{4.13}$$

We take two example cases. The first case has a small initial step size relative to the distance of the starting point to the minimum. The second case has a large initial step size and a close starting point to the minimum relative to the step size. For the first case we take $x_0 = 10.001$ as starting point and a learning rate of $\eta = 0.1$. All numerical minimizers have a maximum of 400 iterations. For ADAM, we choose the recommended hyperparameters from [43]. We did run all tests with different hyperparameters to check if the conclusions in this section are hyperparameter dependent. We could not find any parameter set for ADAM that invalided our findings. For the second case we choose $x_0 = 1.001$ and the initial step-size as $\eta = 1.5$. Figure 4.2 shows our results. We only display the first 60 iterations for clarity in the plot for the second case.

In both cases, our method converges as one of the fastest. The traditional subgradient method converges depending on the chosen a-priori step size rule. While the algorithm with constant step size converges in the first case, the step size is too large in the second case. The opposite is true for a diminishing step-size rule, which decreases too fast in case one while it does converge in case 2. The last method, ADAM, behaves similarly to the constant step-size rule as it minimizes the function until it reaches a stable state iterating between two values. Repeating the numerical minimization for cases 1 and 2 with random starting points, we still observe the same behavior

This numerical excursion shows that the adaptive step size makes the subgradient method faster to converge while reducing the influence of the initial step size.

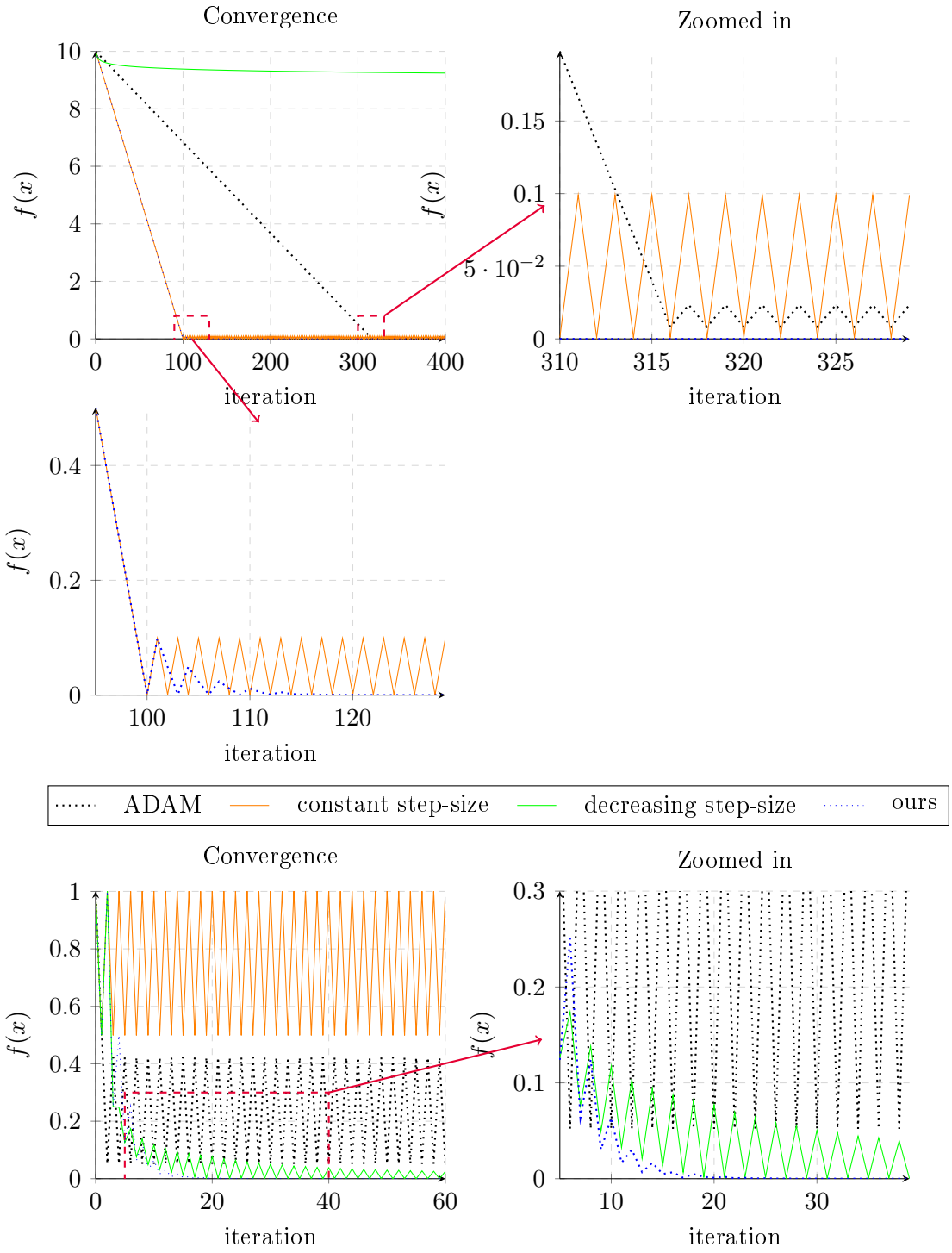

Figure 4.2: Comparison of convergence of different subgradient methods in two example cases. **Top:** Case one with small initial step size and x_0 far away from minimum. **Bottom:** Case two with large step size and x_0 close to minimum.

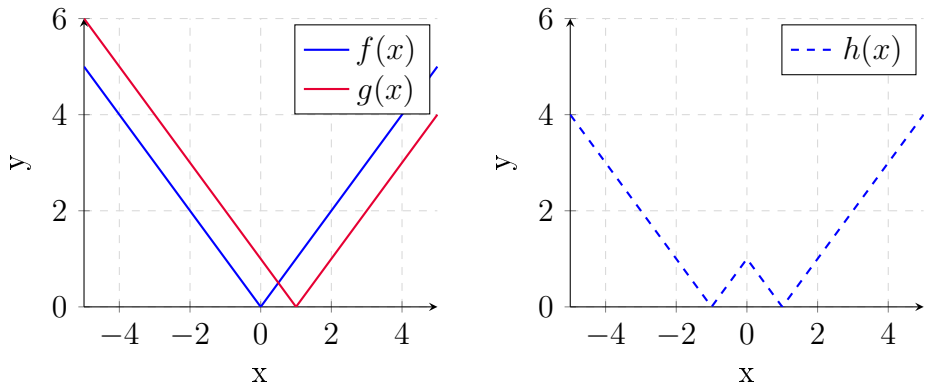

Figure 4.3: Example of the composition of two non-differentiable convex functions. Two convex functions (left) and their (non-convex and non-differentiable) composion (right).

4.4.3 Calculation of Subgradients

To calculate the subdifferential of different, forward models and penalty terms for the Tikhonov functional, some calculation rules can be used. The addition of subdifferentials is anologus to the addition of gradients.

PROPOSITION 57 (*Addition of Subdifferentials*)

Let M be a convex subset of a real Banach space X and $f, g : M \to \mathbb{R}$ be proper convex functionals. Assume that there is a point $x_0 \in \mathcal{D}(f) \cap \mathcal{D}(g)$ where f is continuous, then

$$\partial(f + g)(x) = \partial f(x) + \partial g(x)$$

holds for all $x \in M$.

For a proof see [90], Theorem 47.B. While addition and scalar multiplication follow the same rules as for derivatives, a general chain rule does not exist. A small example illustrates the challenges and limits of deriving a chain rule. We examine the two functions

$$f(x) = |x|$$
$$g(x) = |x - 1|$$

defined for $x \in \mathbb{R}$, each of which is nondifferentiable and convex on the whole domain. The composition $h(x) := g(f(x))$ is non-differentiable as well but not convex on any subset in $\mathbb{R}$ that includes 0, e.g. $[-1, 1]$. Therefore, the subdifferential of h does not exist at 0. Both functions, f and g, have subdifferential on the whole domain $\mathcal{D} = \mathbb{R}$. A general chain rule for subgradients, equivalent to the one for gradients, can not be derived, as illustrated by this example. In figure 4.3 the described situation is shown.

We can, however, derive a chain rule for linear operators. This is stated in the following proposition.

PROPOSITION 58 (*Chain rule linear operator*)

Let X and Y be real Banach spaces, $A : X \to Y$ be a bounded linear operator and $f : Y \to \mathbb{R} \cup \{\infty\}$ be convex and lower semi-continuous. If there is a point $A\overline{x}$, where f is continuous and finite, then for all $x \in X$ it holds that

$$\partial(f \circ A)(x) = A^* \partial f(Ax).$$

This proposition directly follows from corollary 16.72 in [7], which is proven in the same book.

Many examples have a linear forward operator in this work and use a TV norm as regularization. As a result, the subgradient can be directly calculated by using a chain rule for subgradients as well as additive and max rule. For $p = 1$

$$\partial \left\| Au - v^\delta \right\|_{1,\varepsilon} = A^* \text{sign}\left(\max\left\{\left|Au - v^\delta\right| - \varepsilon,\ 0\right\}\right) \tag{4.14}$$

where A^* is the adjoint operator of A. Since the TV-norm is a special case with A being the derivative operator and $\varepsilon = 0$, the same results can be directly applied for the regularization $\mathcal{R}$.

4.5 Comparison of methods

This chapter introduced different numerical methods, and we developed an adaptive step size for the classic subgradient method. The following section compares these numerical methods by minimizing various types of objective functions. Finally, we use these comparisons to look at performance and why we developed a new subgradient method for our applications.

In [5, 21, 55] various functions with different properties for testing numerical minimization methods are presented. We choose a convex function with a unique global minimum, a smooth but non-convex function with one unique global minimum but a multitude of local minima, a non-smooth convex function, and a linear inverse problem for numerical testing.

Smooth convex functions are differentiable and have a unique global optimum which is also the only local optimum. The test function for the smooth convex case is

$$f(x_1, x_2) = 5x_1^2 + 6x_1x_2 + 5x_2^2. \tag{4.15}$$

Smooth but non-convex functions can have more than one local minima and these minima do not need to be a global minimum. We use the following function

$$f\left(x_1, x_2\right) = 0.2 + \sum_{i=1}^{2} \left(\left(x_i - 2\right)^2 - 0.1\cos\left(6\pi\left(x_i - 2\right)\right)\right) \tag{4.16}$$

as an example of a non-convex smooth function with multiple local minima but one unique global minimum. Figure 4.4 shows this function on the input space $[-1,\ 1]^2$.

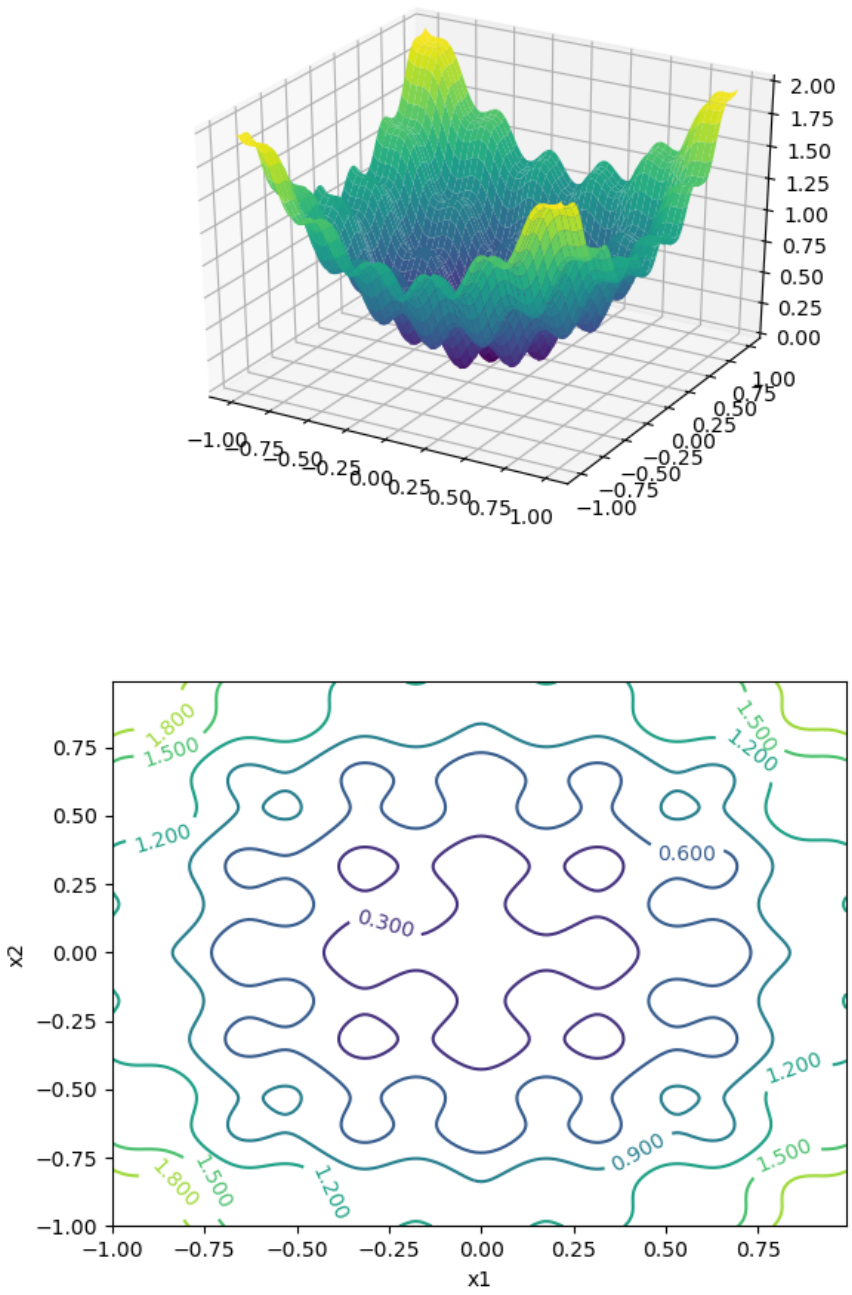

Figure 4.4: Surface plot of a non convex function (4.16) with multiple local minima.

Non-smooth convex function are not differentiable and can not be minimized by using a standard gradient descent method. We choose a non-smooth variant of the Rosenbock function. The function is defined as

$$f(x_1, x_2) = 2\,|x_2 - |x_1|| + |1 - x_1|\,, \tag{4.17}$$

which has a unique minimum at $x = (1, 1)$.

We use the following parameters for the convex and non-convex examples. The CGO has the starting area $[-4, -4] \times [4, 4]$ with a population size of 25 and 800 iterations. The permutation chance is set to 0.2. For the PSO algorithm, we set the starting area to $[-4,\ -4] \times [4,\ 4]$, the population size to 25, and 800 iterations. For the internal weights we choose $\{0.5,\ 0.2,\ 0.3\}$. For ADAM we set the starting point randomly as $x \sim U([-4,\ -4],\ [4,\ 4])$, $\beta_1 = 0.1$, $\beta_2 = 0.9$, and the maximum number of iterations to 2000. For the early stopping criteria on the gradient, we choose 10^{-9}. We provide the analytic (sub-)gradient for the algorithm to use. Each minimization is repeated 100 times to ensure that the comparison is independent of random starting points.

We take the linear inverse problem

$$Au = v$$

with $u, v \in \mathbb{R}^{100}$, and $A \in \mathbb{R}^{100 \times 100}$ an upper triangle matrix. The coefficients of the matrix A are zero if in the lower triangle or 0.01 if on the main diagonal or in the upper triangle. The numerical solvers minimize the functional

$$J = \left\| Au - v^{\delta} \right\| + \alpha \|u\|^2$$

with $\delta = 0.001$, $\alpha = 0.001$ to solve the inverse problem. The noise on the measured data follows a normal distribution $N(0, \delta)$.

We summarize all numerical results in table 4.2. All methods are implemented in python 3.6 and only use Numpy as an additional module and means of optimization. The three non-gradient-based methods perform well on the low-dimensional problems. The Genetic Algorithm performs the best but is the slowest algorithm. As seen in the table, the linear inverse problem test case, all three non-gradient-based methods are slow for large-scale problems. ADAM is faster than the other three methods, but it does not find a global minimizer reliable in the non-convex and non-smooth test cases. The results for the non-convex test case are as expected for ADAM since it does rely on local (sub-)gradients and can therefore get stuck at local minima. We did not expect that ADAM would have these issues solving the non-smooth function. This function has a unique global minimum and no other local minima. We run several experiments with different parameters for ADAM but did not find any better hyper-parameter set for it. For an overview of the various numerical methods discussed, we list the algorithms in table 4.1 with their properties for solving optimization problems.

Solver	p	q	F	Global	Large input space
Nelder Mead	≥ 1	≥ 1	non-linear	$\times$	$\times$
Genetic Algorithm	≥ 1	≥ 1	non-linear	✓	$\times$
Particle Swarm	≥ 1	≥ 1	non-linear	✓	$\times$
Simulated Annealing	≥ 1	≥ 1	non-linear	✓	$\times$
ADAM	≥ 1	≥ 1	non-linear	$\times$	✓
SVM Solver	$1, 2$	$1, 2$	linear	$\times$	✓
Proposed Method	≥ 1	≥ 1	non-linear	$\times$	✓

Table 4.1: Summary of different numerical non-smooth solvers and their properties.

Function	dim	Solver	min f	max f	mean f	min $\|x - x^*\|$	max $\|x - x^*\|$	mean $\|x - x^*\|$
Convex	2	GA	0.0	8.697e-07	2.000e-08	1.476e-10	2.153e-4	7.509e-06
		PSO	0.0	0.0	0.0	1.302e-11	7.064e-10	4.032e-10
		SA	1.344e-5	3.759e-2	7.880e-3	8.465e-4	4.557e-2	1.826e-2
		ADAM	1.773e-10	8.875e-07	1.814e-07	9.417e-06	6.661e-4	2.446e-4
Non-Convex	2	GA	0.0	0.0	0.0	0.0	0.0	0.0
		PSO	6.713e-132	7.063e-112	1.412e-113	1.279e-66	1.876e-56	3.753e-58
		SA	7.637e-06	1.500e-3	4.464e-4	1.002e-3	2.647e-2	1.046e-2
		ADAM	0.227	10.716	6.211	0.164	3.240	2.203
Non-Smooth	2	GA	0.0	1.687e-05	3.445e-07	0.0	2.385e-05	4.828e-07
		PSO	0.0	1.158e-1	3.714e-3	0.0	1.638e-1	5.253e-3
		SA	3.976e-3	8.110e-2	3.321e-2	3.381e-3	9.235e-2	2.978e-2
		ADAM	2.027e-1	4.591	1.453	1.754e-1	5.112	1.711
Linear I.P.	100	GA	5.0e-2					
		PSO	3.038e-1	4.883e-1	4.007e-1	1.932e-1	2.766e-1	2.414e-1
		SA	2.877e-1	3.478	1.549	1.424	15.081	5.202
		ADAM	4.5983e-2	4.703e-2	4.654e-2	1.852e-2	2.325e-2	2.055e-2

Table 4.2: Comparison of different numerical non-smooth solvers for different test functions. Each solver was tested with 50 different initial values.

4.6 Applications in Image and Signal Processing

The following section presents examples for reconstructing blurred and noised data. We do not only consider noised data but noised or partly known forward operators. Our method with tolerance in the discrepancy term is compared to other methods that are used to solve the same problems. The first example is a toy example in 1D, which we use to compare our method with another approach introduced in [47]. The toy example is taken from the same publication and focuses on linear inverse problems with imperfect forward models. The second example is motivated from a real application. This allows for a good impression of how the method can be applied but since real data is used, a qualitative statement is more difficult to achieve. In this example, we reconstruct height images from noisy and blurred height images. Height images, also referred to as depth images are similar to regular images, but they have distance information in each pixel instead of color information. They are used for 3D measurements and obstacle detection.

4.6.1 Deblurring and Denoising with Imperfect Forward Operators in 1D

Let u be a piecewise-constant signal that is deblurred through convolution with a Gaussian blurring kernel

$$\Phi(t;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}.$$

with $\sigma \in \mathbb{R}_+$ and $t \in \mathbb{R}^n$. The true signal is piece-wise constant, hence $\mathcal{R}(u) = TV(u)$, with anisotropic TV-norm for $q = 1$ and isotropic TV-norm for $q = 2$, is chosen as a regularizer. We consider two cases of noise: additive noise in the measurement v and noise on the linear forward operator A. We use three methods to reconstruct the true signal. These methods are minimizing a classical Tikhonov functional with TV-regularization, minimizing a Tikhonov functional with additional point-wise tolerance in the discrepancy term, and the methods from [47], where Korolev and Lellmann introduce a method based on partial ordered Banach spaces to solve inverse problems with noisy or unknown forward operators.

The method from Korolev and Lellmann solves the minimization problem

$$\begin{aligned} &\text{minimize} && \mathcal{R}(u) \\ &\text{subject to} && f^l \leq A^u u,\ A^l u \leq f^u \end{aligned} \tag{4.18}$$

where A^u and A^l are bounds between which the true forward operator exists as well as f^l and f^u are bounds on the data.

The noise on the measurement is assumed to be bounded by $\|v - v^\delta\| \leq \delta$ with $\delta > 0$. We use uniform distributed noise in the following examples.

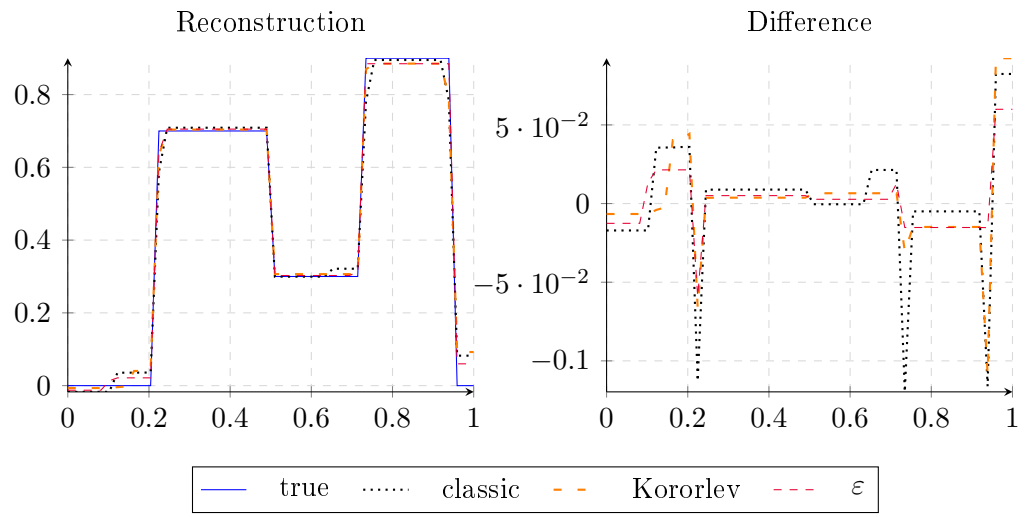

Figure 4.5: Denoising with **exact** operator and **noised** data. Left side: reconstruction of the original signal. Right side: Error in reconstruction.

Figure 4.5 shows an example reconstruction for a noise-free operator but noisy data with uniform additive noise of a noise level of 5%. All three methods can reconstruct the signal almost perfectly. The starting value u_0 was set to $u_0 = v^\delta$ for all algorithms. We find the same results using random starting points.

To add noise to the forward operator A while keeping its properties as an convolution with an appropriated kernel, the additive noise is added on the Gaussian kernel

$$\Phi^\delta(t) = \frac{1}{\| \max\{\Phi(t) + \varepsilon(t), 0\}\|} \max\{\Phi(t) + \varepsilon(t), 0\},$$

where $\varepsilon \sim U\left(\frac{-\delta}{\sigma\sqrt{2\pi}}, \frac{\delta}{\sigma\sqrt{2\pi}}\right)$. The normalization of the operator is necessary for the convolution properties. For the operator in matrix forms, this approach translates to only non-zero elements being affected by the noise. Due to the normalization of the kernel the noised matrix is a convolution matrix again.

In Figure 4.6 an example reconstruction for 10% noise on the operator and 5% additive uniform noise on the data is shown. The reconstruction is less accurate than before, as the overall noise level increased depending on u.

For a more detailed comparison, the above example was repeated with randomized starting values u_0. Each starting value is repeated ten times with random noise using the same noise level. The solutions are shown in table 4.3 and illustrate that the above example is not an unique case, but similar observations can be done for different sampled noise as well. It is important to note that for additive truncated normal noise, both Tikhonov functionals yield significantly better results than the method from [47]. Yuri et al. assume worst-case errors. They do not take the error distribution into account. Therefore the higher density of the error around zero is not exploited.

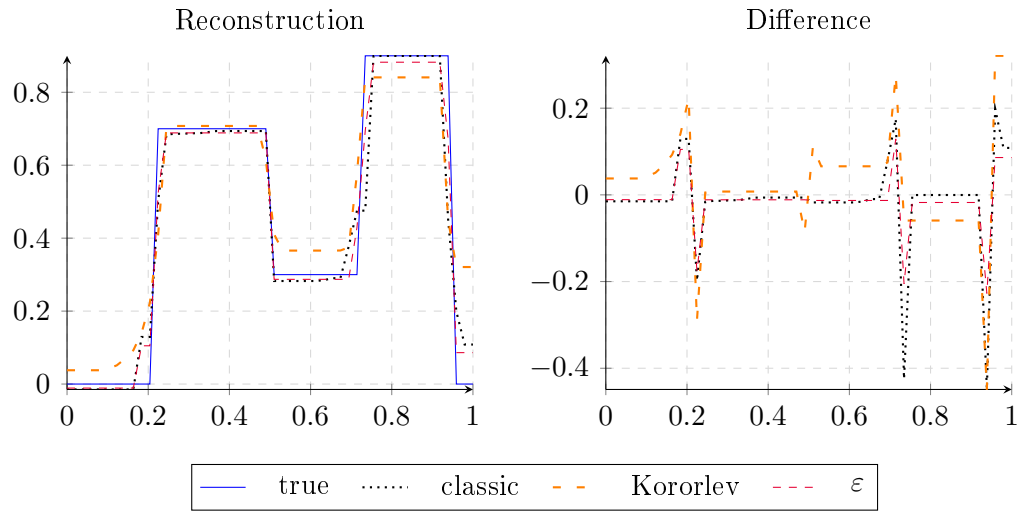

Figure 4.6: Denoising with **noised** operator and **noised** data. Left side: reconstruction of the original signal. Right side: Error in reconstruction.

noise	δ_v	δ_F	Classic	Korolev	ε
uniform	0%	5%	0.3353 ±0.027	**0.1301**±0.022	0.2466±0.027
	10%	5 %	0.2909 ± 0.029	0.7420 ±.0034	**0.2167** ±0.026
normal	0%	5%	0.1053 ±0.024	0.1321±0.023	**0.0699** ±0.031
	10%	5%	0.2174 ±0.034	0.7124 ±.0031	**0.1780** ±0.018

Table 4.3: 1D deconvolution for different noise models and noise on the data and the convolution operator. The difference to the true signal is measured by $\|u - u^\delta\|_2 10^{-2}$ with $p = 2$ and $\mathcal{R}_q = TV_1$. All values show the mean of 100 repetitions with the standard deviation in the set.

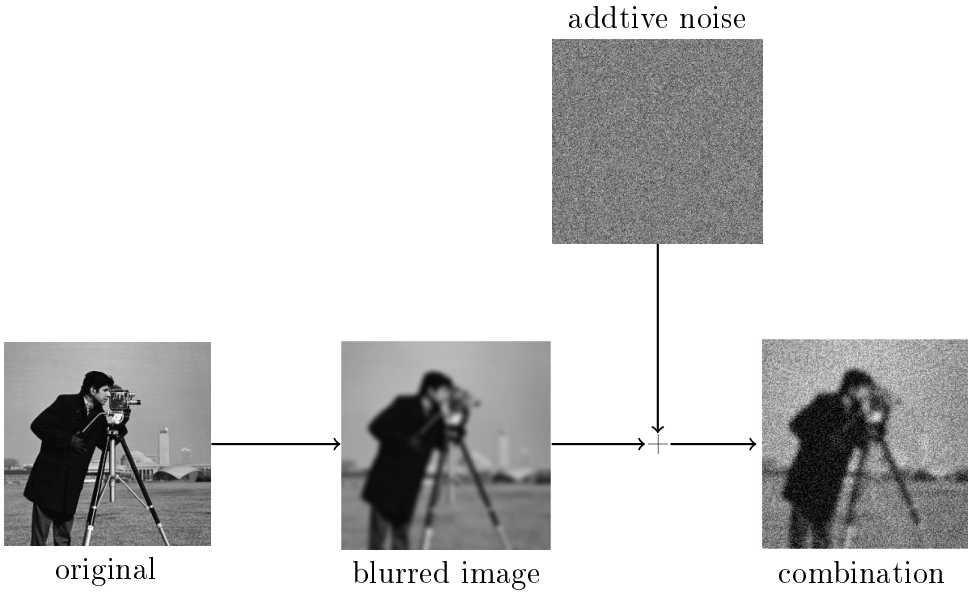

Figure 4.7: Example of blurred and noised image as a result of convolution with Gauss kernel and additive noise error on each pixel.

4.6.2 Deblurring Depth Images

We use real-world data to apply the proposed method to depth images. The experimental data set is the same as used in [34]. It is provided by the Cooperate Research Center 747 (CRR 747) of the University of Bremen. Micro parts in the shape of a cup are cold formed and measured directly on the cold forming machine. A confocal microscope Keyence VK-9700 is used for the measurement. The used materials are copper and aluminum. The images produced by the microscope are depth images. Instead of color information, each pixel holds depth/height information. These images may also hold non-valid values in a pixel and need additional normalization. A short overview of the different techniques and sensors can be found in [35] between pages 256-258. Figure 4.8 illustrates the difference between depth images and regular color images of a 3D scene.

Shape deviation of manufactured micro cups from ideal shape shall be detected with only information from the depth images. Since the images are prone to significant errors, post-processing is necessary. This post-processing aims to eliminate noise while extracting the actual shape. To illustrate the influence of noise on the image, we apply four existing algorithms and outline possible enhancements with our algorithm. We apply all four algorithms to the same image and show the results in figure 4.9. We choose the best meta parameters for each algorithm via optimizing meta parameters if necessary. For this optimization on the hyperparameters, we use a two-step solution. First, Bayes optimization [73] is performed and then locally refined with a convex solver such as the Broyden–Fletcher–Goldfarb–Shanno algorithm [37]. We only apply the second step for hyperparameters in a continuous space.

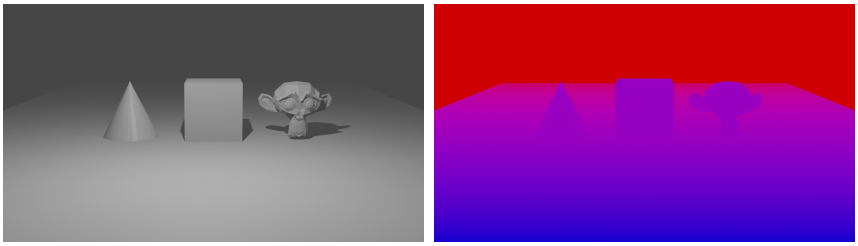

Figure 4.8: Example of a depth image. Left: Rendered image of the scene with point lightsource and shadows. Right: The depth image of the same scene. Blue areas are closer to the camera and red areas are further away. Both images were simulated and rendered using Blender 2.9. The images are best viewed in color.

This method allows for a vast search space while still finding local optimal points.

A convolution with smoothing kernel followed by additive noise models the overall noise in the image.

$$v^\delta = (\Theta(\sigma, \mu) * u)(x) + e(x).$$

Observations on collected data and other experiments in the CRC 747 framework support this model. We can interpret the convolution as the unknown forward operator in our method. We use TV-regularization with $q = 1$ to minimize the assumed error. Since we have an additive error, we choose $p = 2$. Overall the inverse problem is

$$\begin{aligned} F : L_p(\Omega) &\to L_p(\Omega) \\ u &\to \Theta(\sigma, \mu) \cdot u \end{aligned}$$

for the forward model where Ω is a Gaussian kernel

$$\Theta(x; \sigma, \mu) = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^2} e^{-\frac{\|x-\mu\|_2^2}{2\sigma}}.$$

We choose the necessary α and ε by using a generalized L-curve (see [54]), also known as L-shaped hypersurface. Figure 4.9 shows the result. Our method can remove almost all noise on the border of the depth image while keeping and refining the contour of the cup. Since we applied an automatic selection of the meta parameters ε and α, we can automate this method without user input. As a result of real-world data, the images can only be compared subjectively as we are unable to compare the results with some known true values. Nevertheless, the results coupled with the knowledge from the toy experiment in one dimension show the potential of our method.

The standard gradient detection with a Sobolev filter shows the degree of noise in the area around the cup. The image was taken while the cup was on a plane surface. Thus, we can assume that the observed gradients

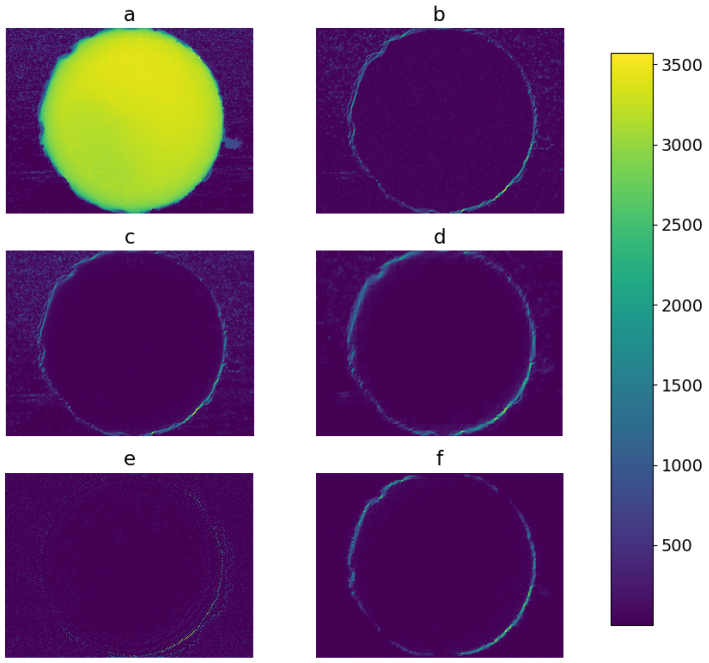

Figure 4.9: Edge detection on 2D images using different techniques. (a) Original noisy image. (b) Sobolev filter on Perona Malik denoised image. (c) Gradients with Sobolev filter. (d) Sobolev filter on Biorthogonal wavelet denoising with Bayes shrinkage denoised image. (e) Canny Edge algorithm. (f) Sobolev filter on our algorithm denoised image.

around the structure must be noise. Applying the *Canny Edge Detection Algorithm* [13] reduces the noise around the cup but is not able to extract the entire contour of the cup. Biorthogonal-wavelet image denoising with Bayes shrinkage method, as described in [29, 19], produces blurred contours. Of the four algorithms, the Perona-Malik smoothing, as introduced in [34], yields the best results. However, it is not easy to automate this method with the appropriate stopping criteria for the Perona-Malik smoothing. Moreover, we noticed that this method can lead to contour blurring, which is an undesired property for the application at hand. The method proposed in our work extracts the contour and reduces noise outside the cup. Furthermore, it allows the extraction and analysis of the shape for defects.

5 Application in Micro Milling

We originally designed the presented mathematical methods in this work to be applied to parameter identification for micro-milling as part of the cooperative research center 747 (CRC 747) at the University of Bremen. In the CRC 747, new methods for cold forming micro parts in micro dimensions, i.e., parts smaller than 2 mm in two dimensions, are studied.

The need to manufacture smaller and smaller components to keep highly sophisticated systems at an acceptable size bears new challenges for manufacturing processes. These challenges are mainly due to size effects [86] which hamstring the transition of methods developed in macro-scale to micro-scale. Milling processes are an example where size effects need to be considered. Figure 5.1 shows an example of dry friction in a mold used for micro cold forming. The surface structure of the material influences the friction of the blank.

In a milling process, small structures on the surface of the finished workpiece are inevitable. They can be ignored for the macro-scale but highly influence the characteristics of a component in the micro-scale. These characteristics are crucial for microstructured devices, or functional tribological surfaces [58, 6, 23]. Understanding the cutting process and influential process parameters in detail allows for identifying parameter sets that produce surfaces structured in the desired way.

Various applications for the mathematical method described in this work exist for micro cold forming processes. They can be divided into two main categories. First, understanding forming process and identifying underlying correlations or causes (innovation speed), and second, enhancing process stability and productivity through parameter identification (process design and control).

We give a few selected examples which focus on micro-milling processes of each field in the following sections.

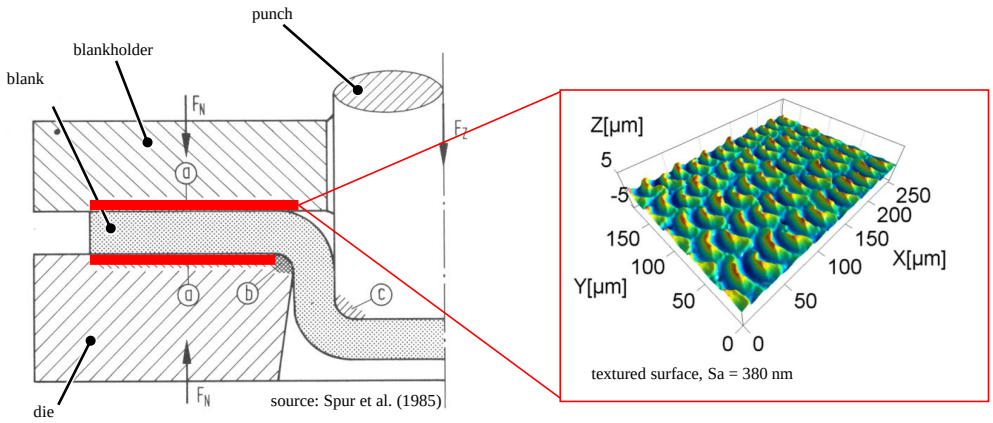

Figure 5.1: Dry friction in a mold used for micro cold forming.

5.1 Technical Background

Milling is a versatile technique from the early 19th century and is used to produce parts that range from tiny to huge. It is applied to various materials and has been steadily improved over time. This work focuses on the application of metal on a micro-scale. We follow [86] in their definition for micro parts. Thus all workpieces are less than 2mm in the sum of all their dimensions, i.e., the sum of height, length, and depth of the smallest bounding box is less than 2mm.

In forming processes, the forming tool is subject to forces and the workpiece. Forces during the process alter the characteristics of the forming tools. These characteristics include geometry and hardness and represent wear on the tool. In addition, the wear of tools alters and affects the characteristics of produced workpieces, especially the surfaces in our application. For example, the shape of the cutter and cutting edge influence the surface quality in micro-milling processes. As wear on the cutter increases, the cutter itself has to be replaced to ensure that the desired properties on the manufactured surfaces are met.

5.2 Identifying Acceleration Points from Positional Data

The first application of the introduced method is modeling and finding the points of acceleration of the cutter from positional data. Milling machines use target positions and speed, and interpolation type (linear, cubic) to control the milling process. These data points are generated beforehand and provided to the machine in a standard binary format. During the process, the machine records feedback of the force on the tool and its current position.

This information is then used to control the whole process, identify failure in advance, and improve the input parameters. In contrast to classical regularization, which still takes place, our goal was to adjust process parameters and adapt the underlying model to reflect the actual process better and find a source for observed anomalies. For this task, it is essential to reproduce the acceleration of the milling drill from its positional data provided by the machine. In micro milling processes, the force on the cutter has a significant influence on the produced workpiece. See [79] for further information on this topic. One possible source is the acceleration of the cutting tool, which can not be measured directly. Instead, an indirect method is used, which is based on the known positional data of the cutting tool.

The first derivative of the traveled distance $s(t)$ is the velocity $v(t)$. The second derivativeof the traveled distance is the acceleration $a(t)$, i.e.

$$\frac{d}{dt}s(t) = v(t)$$
$$\frac{dd}{dtdt}s(t) = \frac{d}{dt}v(t) = a(t)$$

Finding the acceleration from a given distance measurement is a very ill-posed problem due to the compactness of the operator $F : s(t) \to a(t)$. This means that taking the second derivative from $s(t)$ with respect to t is highly inaccurate when accounting for noisy data. Small deviations in the measured distance can lead to unbounded errors in the reconstructed acceleration by taking derivatives. We use the proposed Tikhonov functionals to compensate for the ill-posed nature of the problem.

We need to know the true acceleration to test the reconstruction accuracy. The CNC code does include this information, but we want to reconstruct the true acceleration if it deviates from the time points given in the CNC code. In addition, this example is more of a case study than meant to be applied later. We, therefore, will use artificial data based on real-world observation. This artificial data allows us to know the otherwise known true acceleration and allows for a comparison of accuracy. It was first published in [27] to show the additional regularization properties due to tolerances in discrepancy terms. We will add additional information on the implementation and mathematical structure of the example in this work.

The acceleration $a(t) : \mathcal{D} \subset \mathbb{R} \to \{-1, 0, 1\}$ can only be in one of three states at any time, forward, no acceleration and backward. The number of switches between the three states is not restricted. For the regularization the amount of switches is considered, i.e., sparsity in the time points of changed states is desired. The TV-norm on the signal $a(t)$ will be used as $\mathcal{R}_q$ to match the desired properties. The true solution starts at 0 and switches at $t = 0.3$ to 1 and $t = 0.6$ back to 0. A uniform additive noise with $\delta = 15$ is used on the positional data $d(t)$. To solve the minimization problem a greedy approach is selected. For each number of possible switches $i \in \mathbb{N}^+$ the minimization problem

$$\min_{p \in [0,1]^i} D(f(x; p) - y) + \mathcal{R}(f(x, p))$$

is considered, where

$$f(x;p) = \int_{\mathcal{D}} a_p(t)\, dt \tag{5.1}$$

$$a_p(t) = \begin{cases} 1, & \text{if } i \text{ is odd, defined by } p_i < t < p_{i+1}, \\ 0, & \text{otherwise} \end{cases} . \tag{5.2}$$

We consider the switch points p_i to be shorted in ascending order.

We increase the complexity of the function f step-wise and take the first local minima in i as the best solution overall. Thus, if we find a solution at $i = c$, we computed $c + 1$ minimizers. An early stopping criteria is fulfilled if $f(x;p) = 1$ almost everywhere.

The classical solution with no tolerance is $\{0.2888, 0.5801\}$ and has an error of $5.5678 \cdot 10^{-2}$ in the L_2-norm. This reconstruction could not be improved with a different value of α. Minimizing $J^{2,1}_{\delta,.1,5}$, where

$$J^{p,q}_{\delta,\alpha,\varepsilon}(u) = \frac{1}{p} \left\| F(u) - v^\delta \right\|^p_{L_p,\varepsilon} + \alpha \|u\|^q_q, \tag{5.3}$$

finds $\{0.2421, 0.2424, 0.2991, 0.6001\}$ as switch points and has an error of $1.4142 \cdot 10^{-2}$ in the L_2-norm. The regularization parameter was set to the same value as for the Tikhonov regularization without tolerance in discrepancy. The reconstruction and data is given in fig. 5.2.

While the classical approach finds the right amount of switches, it has a more significant error than the Tikhonov functional incorporating tolerances. The simulation at hand was repeated 100 times with random noise, and the additional switch did not always appear. However, the overall error of our method was always smaller than the reconstruction based on the standard Tikhonov regularization. Thus the already good reconstruction by the classical regularization can be further improved by incorporating a tolerance in the discrepancy term. It is important to note that the restriction in the solution space through the construction of $f(x;p)$ to be discrete in the number of switches is another way of regularization.

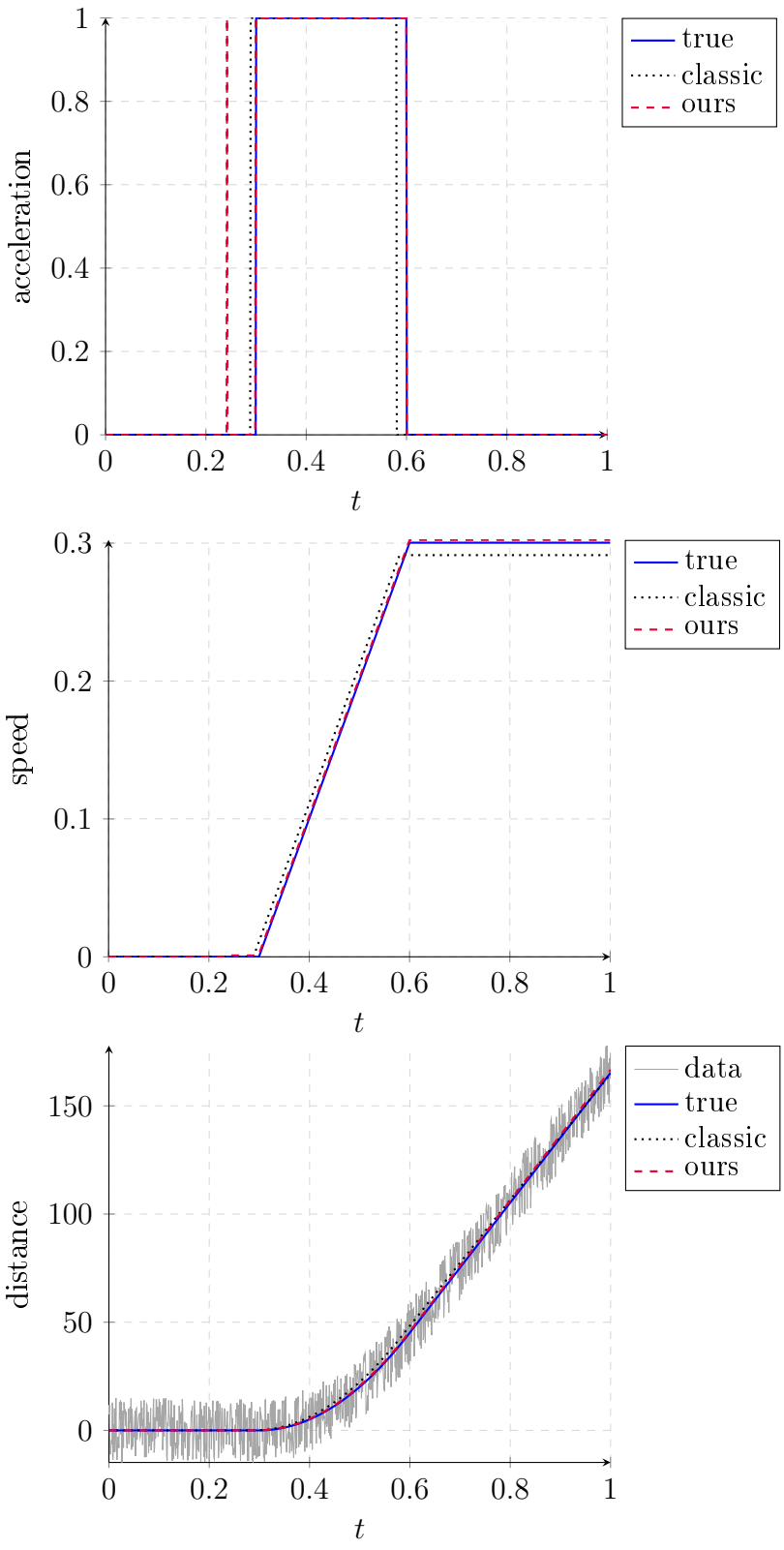

Figure 5.2: Reconstruction of acceleration from positional data. Using classical Tikhonov functional with TV-regularization and Tikhonov functional with tolerance in the discrepancy term in addition to TV-regularization.

5.3 Influence of the Wear of Cutting Tool on Cutting Process

In this section, we outline a process-model for micro-milling and model the wear on the tool. We further provide detail on parameter selection with respect to the wear and how to integrate the theory of inverse problems.

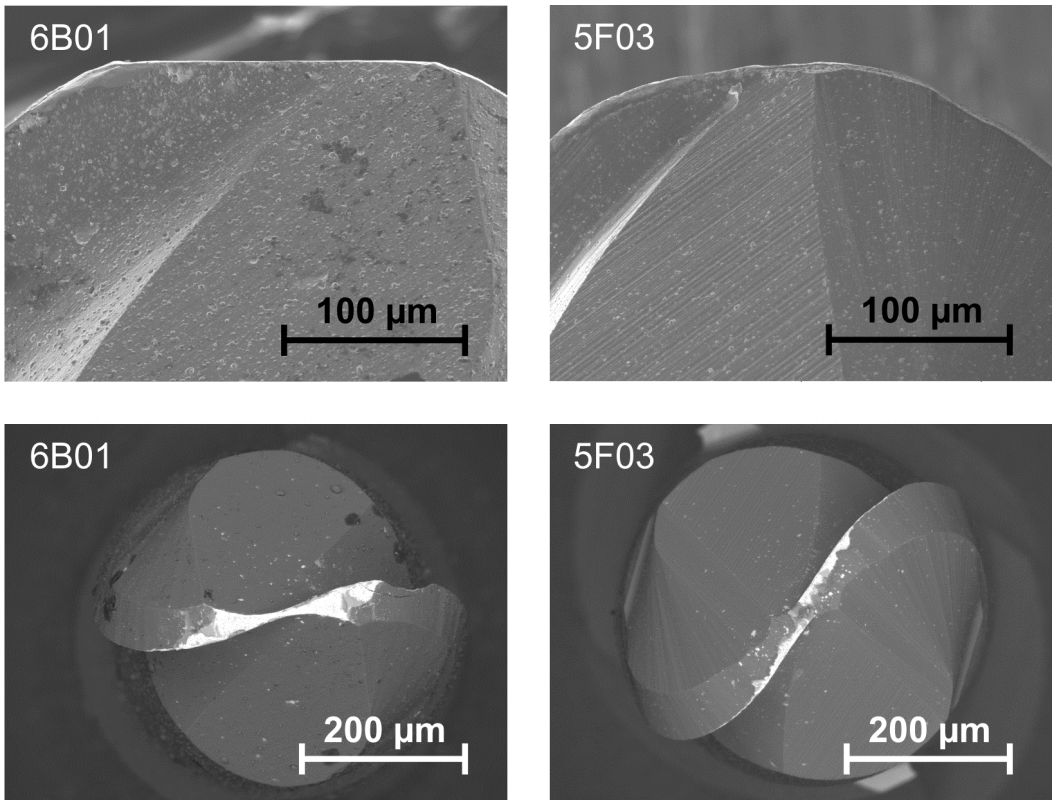

Figure 5.3: SEM pictures of used cutting tools. The left column shows an even wear on the cutting tool. The right column shows an uneven wear on the cutting tool. The white areas on the bottom row are wear.

5.3.1 Process model

We model the process in three steps. In the first step of modeling, we consider the kinematic of the process. We describe the kinematic of a milling cutter by using a composition of the rotational and translational motion of the tool. This composition describes how the tool rotates around itself and moves along a given path. A dynamical force model is used to account for dynamical changes occurring during the process. We utilize a system of ordinary differential equations to represent this dynamical force model. The model is presented in details in [60] and [83]. Moreover, we assume that the reader is familiar with

the kinematics of the micro-milling process. If otherwise, we refer the reader to [3] and [79].

To model the final surface, additional information like the shape of the cutting tool, material and size of the workpiece, and used material are needed. Additionally, we included the material removal mechanism. The minimum chip thickness has to be considered for ball end micro-scale machined surfaces. The minimum chip thickness results from the chip formation, which is divided into three parts see [83]. Below a certain minimum cutting depth, the material is elastically deformed, and thus no material is removed, i.e., no chip is formed. Above a certain cutting depth, the tool completely removes the material, and the chip thickness equals to the cutting depth. The third stage describes the transition between the former two stages. Since the cutting depth is different at all points of the milling tool and only cutting depths smaller than the tool radius are used for the last milling step, all three stages are constantly present. The material removal process results in microstructures on the produced surface, depending on the process parameters.

The forward model generates a detailed map of the produced surface. Thus it can be used to compare the characteristics of surfaces from different input parameters. However, the comparison of two-dimensional surfaces is rather complicated and not always practical because the desired properties might not be available in the form of a two-dimensional surface. For these specific reasons, different ways to compare surfaces are needed.

5.3.2 Kinematic Process Model

This work focuses on the kinematics within the process and neglects the forces and deformations on the cutting tool during the milling process. This results in a less exact simulation, but the error is small compared to the influence of the wear on the tool [82]. For more complex geometries, the influence of deflection and force on the cutting tool is more severe, and the full model should be applied in this case. The simplified model focuses on the wear and geometry of the cutting tool and its influence on the surface generation process. It is therefore ideal to focus on specific properties of the milling process. In addition, the workpiece and cutting path in this work is on one plane. The implementation of the forward model is based on [83] done by Dr. Vehmeyer in MATLAB. We expand the model implementation with a wear model for the cutting piece. All parameter identification and inverse problem solutions have been implemented for this work but use the extended model as the forward operator.

In the following, the modeling process is separated into five different sections, which are explained in more detail before summarizing the whole model in one. The first section lists and explains all process parameters. The second section discusses the tool path and the tool geometry. The third section focuses on material removal and chip formation. The fourth section examines wear on the cutting tool and it is modeled. Finally, the last section covers the

symbol	description
$a_e \in \mathbb{R}^+$	offset
$v_f \in \mathbb{R}^+$	feed speed
$f_0(\phi) \in C(\mathbb{R})$	tool geometry without wear
$\theta(x, y) \in L^2$	surface before cutting process
$r \in \mathbb{R}^+$	radius of the tool
$l \in \mathbb{R}^+$	length of the tool
$\Omega \in \mathbb{R}^2$	area of the tool that has been affected by the cutting process
$\rho(\phi) \in C^2(\mathbb{R})$	function for describing curvature of the beval edge

Table 5.1: Parameters

classification of the generated surfaces and strategies for comparing surfaces.

Each of the listed topics is further explained before the whole model is summarized and presented as one.

Model Parameters The model in this work has two parameters for the tool path. The offset a_e and the feed speed v_f. These are the only two changeable parameters for the tool path in this work. In general, the tool path can be modeled with a broader description, see for example [83], where the parameters are time-dependent.

Besides parameters for the tool path, the model has parameters for the tool geometry, surface before the cutting process, tool radius, tool length, and tool geometry. All parameters are listed in the table 5.1. These parameters are not time-dependent and are fixed for one simulation. The space of all fixed parameters during a simulation is

$$\mathcal{P} = C(\mathbb{R}) \times L^2 \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^2 \times C^2(\mathbb{R}).$$

Cutting tool The geometry of the cutting tool is described by $f_0 \in C[0, \frac{\pi}{2}]$. This work uses a ball-end cutting tool in the simulations. For a ball-end cutting tool, f_0 forms a half-circle around the point $p = (p_x, p_y, p_z) \in \mathbb{R}^3$. We model the cutting edge as a one-dimensional curve in a three-dimensional space, even if the cutting tool itself is three-dimensional. We account for the thickness and two-dimensional shape of the cutting edge when calculating the material removal. The ideal cutting edge of a ball-end cutting tool is

$$f_0(\phi) := R_z(\phi)(r\cos\phi,\, 0,\, r\sin\phi - l + r)^T, \quad \phi \in \left[0, \frac{\pi}{2}\right]$$

where l is the tool length and r is the ball radius. The function

$$R_z(\phi) : \mathbb{R} \to \mathbb{R}^{3\times 3}$$

produces a rotation matrix around the z-axis depending on the value ϕ. For small cutting depth (e.g. 15μm), the cutting edge beval $\rho(\phi)$ is small and can be approximated with a constant value according to [83] page 2.

Tool path The tool path describes the extrinsic tool movement relative to the workpiece. It is a continuous curve $\varphi : \mathbb{R} \to \mathbb{R}^3$ which is almost everywhere differentiable. A set of control points usually defines this curve. The final curve is then generated through interpolation between the control points. If feed speed, rotation, and parameters are provided in the tool path, the dimension of the tool path does increase accordingly, $\varphi : \mathbb{R} \to \mathbb{R}^n$, where $n \in \mathbb{N}_+$ is the number of parameters.

The machine physically limits the tool path. These boundaries can be added as constrains to the model. For more information about the constrains of the machines in this thesis, we refer the reader to [1].

In the example applications in our work, we apply parallel cutting path with distance $a_e \in \mathbb{R}^+$. If the cutting tool passes over the whole workpiece along one cutting path, it switches to the next path. These paths can be traversed in changing directions, e.g., up and down, or in uniform direction, e.g. up and up.

In addition the feed forward speed $p_f > 0$ of the cutting tool along the tool path is defined via $v_f \in \mathbb{R}^+$. The formula of the parameterized model for $t \in [0,\ 2c\frac{l+a_e}{v_f}]$

$$\varphi(t; a_e, v_f) = \begin{cases} (\varphi_x, \varphi_y + v_f t) & : t \in \left[0,\ \frac{l}{v_f}\right] \\ (\varphi_x + v_f(t - \frac{l}{v_f}), \varphi_y + l) & : t \in \left(\frac{l}{v_f},\ \frac{l+a_e}{v_f}\right] \\ (\varphi_x + a_e, \varphi_y + l - v_f(t - \frac{l+a_e}{v_f})) & : t \in \left(\frac{l+a_e}{v_f},\ \frac{2l+a_e}{v_f}\right] \\ (\varphi_x + a_e + v_f(t - \frac{2l+a_e}{v_f}), \varphi_y) & : t \in \left(\frac{2l+a_e}{v_f},\ 2\frac{l+a_e}{v_f}\right] \end{cases}$$

where $\varphi(0) = (\varphi_x, \varphi_y)$ defines the starting point for $t = 0$ and $l \in \mathbb{R}^+$ is a fixed length defined by the length of the workpiece. For $t > 2\frac{l+a_e}{v_f}$

$$\varphi(t; a_e, v_f) = \varphi\left(t - 2c\frac{l + a_e}{v_f}\right) + (2c \cdot a_e, 0)$$

with $c = \lfloor t\frac{v_f}{2(l+a_e)} \rfloor$. An illustration of the described model is given in figure 5.4.

The operator

$$\begin{aligned} T : \quad (\mathbb{R}^2)^+ &\to C(\mathbb{R}, \mathbb{R}^3) \\ (a_e, v_f) &\mapsto \varphi(t) \end{aligned}$$

describes the movement of the cutting tool center $\varphi(t)$ based on the given parameters $a_e,\ v_f$. With the translation on the curve $\varphi(t)$, the cutting tool simultaneously rotates around its own center. This rotation is modeled with $R(t, p)$ around the point $p \in \mathbb{R}^3$. The angle of rotation h depends on the time t. The cutting tool rotates around its center while simultaneous translating along the tool path $\varphi(t)$. This self-centered rotation is modeled with a rotation $R(t, p)$ along the center axis of the tool.The rotation angle depends on t. The

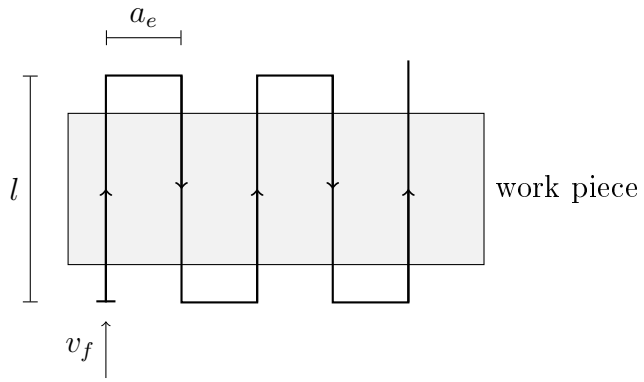

Figure 5.4: **Example cutting path $\varphi(t)$ of cutting tool** on a rectangular workpiece, displayed in top view.

cutting edge f_t of the cutting tool at the time t is calculated through the following model

$$\begin{aligned} f_t(\phi) &= R(t, (T(v_f, a_e)(t)) f_0(\phi) \\ &= R(t, \varphi(t)) f_0(\phi) \end{aligned}$$

where $f_0(\phi)$ is the cutting edge at the time $t = 0$. Figure 5.5 illustrates the rotation of the cutting edge.

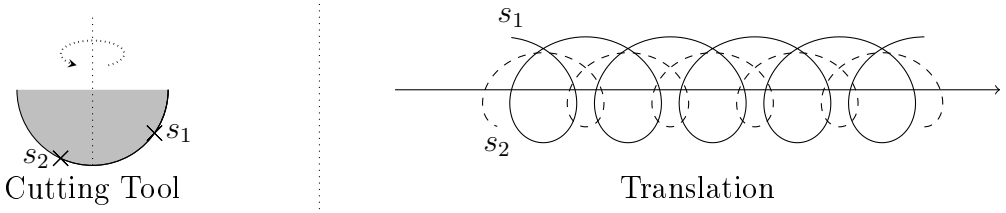

Figure 5.5: **Cutting Path of ball edge micro miller**. On the left is the 2D profile of a ball edge micro miller. On the right the trajectory of two points in the cutting edge.

Surface generation The contact area between the workpiece and cutting tool material is deformed and removed from the workpiece. This process alters the surface of the workpiece as well as the cutting tool. Changing the surface of the workpiece is desired, while the change of the cutting tool is undesired and considered wear.

The moving cutting tool removes material in contact with the workpiece and alters its surface. Only areas that are swept by the cutting tool are affected.

We define the sweep-volume, that is, the trace of the cutting edge as

$$\varrho(S) = \int_S f_t(x, y, z)dt.$$

We define the image of $\varrho(S)$ by the set of points $(x, y) \in \mathbb{R}^2$ that are covered by $f_t(x, y, z)$ for any $t \in S$. We can write this set as

$$Im(\varrho(S)) := \left\{(x, y) \;\middle|\; (x, y, z) = f_t(\phi)\ \forall t \in S \text{ and } \forall \phi \in \left[0, \frac{\pi}{2}\right]\right\}.$$

The material removal and the generated surface is described by the difference between the workpiece surface and the sweep volume $\Theta(x, y)$ calculated as

$$\Theta(x, y) = \min_{t \in S}\{\varrho(S), \theta(x, y, z)_0\},$$

for all $(x, y) \in \mathcal{D}(\varrho(S))$. The surface of the workpiece is the union of the unaffected surface area and $\Theta(x, y)$.

So far we did not consider the chip formation and removal in the model. If the cutting depth is too small, some workpiece material is only deformed and not entirely removed by the cutting edge. Since we look at the process in micro-scaled, the cutting depth cannot be ignored. We also have a ball edge cutter and can't assume an even cutting depth at every point of the tool edge.

We model the material removal by dividing the chip formation into three parts, as done by Amacheron and Mativenga in [4]. These three parts are divided by the two constants $h_1, h_2 \in \mathbb{R}_0$ with $h_1 < h_2$. If the cutting depth h is below the threshold h_1, no material is removed, and only elastic deformation takes place. The transition phase between full chip formation and elastic deformation is $h_1 \leq h \leq h_2$. In this transition phase, elastic deformation and partly chip formation are present. The whole material is removed without any elastic deformation for $h > h_2$. Figure 5.6 shows the three phases.

We extend the surface generation with the minimal chip thickness function $\rho(\delta z)$ and calculate the surface on $(x, y) \in \mathcal{D}(\varrho(S))$ via

$$\Theta(x, y) = \theta(x, y, z)_0 - \rho(\Theta(x, y) - \min_{t \in S}\{\varrho(S),\ \theta(x, y, z)_0\}).$$

Since the minimum chip thickness needs the current height of the surface at the moment of cutting process, we need to calculate the material removal dependent on t. It is important to note that $\theta(x, y, z)_0$ is the surface before the tool changes it at the time t. During the process, the cutting tool can pass the same surface area multiple times, the removed material is different each time depending on the chip thickness in each instance.

5.3.3 Surface Characterization

A measurement or characterization of surfaces is needed to classify and evaluate the outcome of a milling process. The German Institute for norms

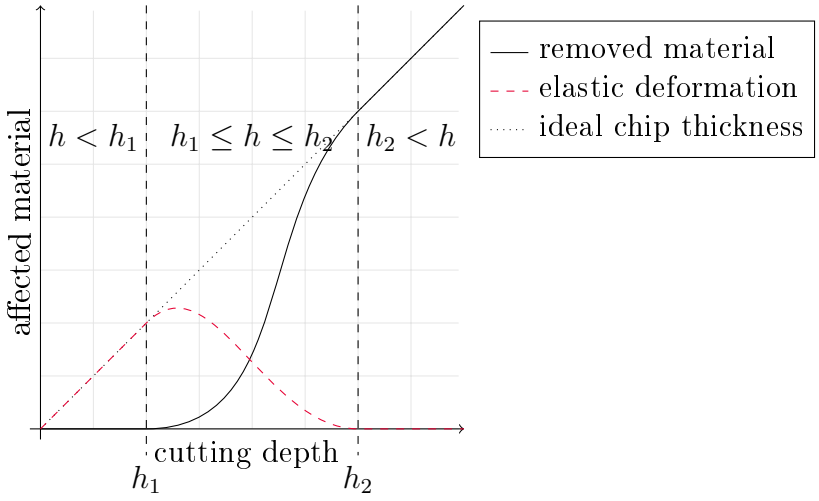

Figure 5.6: Concept of the minimum chip thickness with three phases of chip formation and the continuous truncation function of the surface generation model (Source[84]).

Order	Description	Schematic
1.	Shape deviation	
2.	Corrugation	
3.	Roughness	
4.	Roughness	

Table 5.2: **First four levels of surface deviation** according to DIN 4760:1982-06.

(Deutsches Institut für Normierung [DIN]) defines six types of surface deviations from a predefined ideal surface. These deviations are categorized into six levels. Roughness parameters further specify the deviations in levels three and four. In principle, each level defines a more nuanced, more local deviation. Therefore, if multiple levels of deviation are present, each level should be filtered out via convolution before calculating the characteristics value of the following level. As a result, each level is more complex, and the support of the convolution kernel smaller. The first four types of deviations as defined in DIN 4760:1982-06 are listed in the table 5.3.3. This work focuses on deviations of levels three and four, and we assume that no level one or two deviations exist as these deviation are to coarse to be detected in our application examples.

The roughness of a surface is defined by the following 1D-Parameters in *DIN EN ISO 4287*:

- arithmetic mean roughness R_a

$$R_a := \frac{1}{l_r} \int_0^{l_r} |z(x)| dx$$

- quadratic mean roughness R_q

$$R_q := \sqrt{\frac{1}{l_r} \int_0^{l_r} z^2(x) dx}$$

- averaged roughness depth R_z

$$R_z := \frac{1}{5} \sum_{i=1}^{5} R_z(i)$$

  where $R_z(i)$ is defined as

$$I_i = \left[(i-1)\frac{l_r}{5},\ i\frac{l_r}{5} \right]$$

$$R_z(i) := \max \{z(x) \mid x \in I_i\} - \min \{z(x) \mid x \in I_i\}$$

- maximum roughness depth R_{max}

$$R_{max} := \max\{R_z(i)\}$$

- roughness depth R_t

$$R_t := \max \{z(x) \mid x \in [0, l_r]\} - \min \{z(x) \mid x \in [0, l_r]\}$$

For surfaces without any level 1 or 2 deviation, R_t is identical to the Peak-to-Valley value of the surface. The parameters R_a and R_q depend on a zero height level. All other parameters are independent of any reference height.

In practice, the above characteristics may not be sufficient. They fail to account for the local distribution of peaks, lack information about the height distribution vertically, and do not account for material distribution between peaks and valleys. The Abbott-Firestone curve provides more information about the vertical height distribution. This curve is also referred to as the bearing area curve. In figure 5.7 a visualization for clarification is provided. The Abbott-Firestone curve $f : [0, 1] \to \mathbb{R}$ is defined as

$$f(p) = (P[O(\Omega) \geq c])^{-1}$$

where $\Omega \subset \mathbb{R}^2$ and $O : \mathbb{R}^2 \to \mathbb{R}$ describes the surface. The Abbott-Firestone curve is the quantile density function of the surface profile's height from its mathematical properties. This function is not unique and needs a reference height level. The DIN-Norm defines the zero height as $f(1) = 0$. Through this

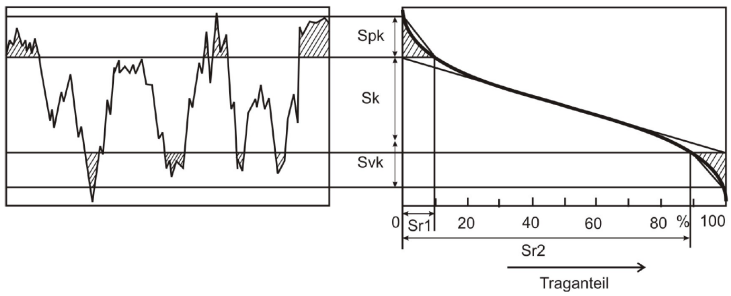

Figure 5.7: **Schematic of a Abbott-Curve** and its calculation (Source [?]).

definition, each surface has a single Abbott-Firestone curve. The converse does not hold, which means multiple surfaces can have the same Abbott-Firestone curve.

We use the Abbott-Firestone curve to characterize a surface and define a desired ideal surface. It includes all necessary information from the DIN-Norm and its parameters, and includes more information about the height distribution. Furthermore, for large enough surfaces and surface patches, the curve is translation invariant, i.e., we can compare two surface patches, and small translations in the surface window patch do not alter the curve significantly. This invariance is a desired property for our usecase in micro-milling. In addition, it is more feasible to define an ideal surface through its desired properties, which can be used to calculate a matching Abbott-Firestone curve.

5.3.4 Comparison of Surfaces

One way to compare surfaces is to divide deviations into six categories as defined by DIN: 760:1982-06 from the Deutsche Institut für Normierung (DIN Standards). Of special interest are deviations of the third and fourth categories describing the roughness of a surface. This is further specified in DIN EN ISO 4287, where so-called one-dimensional characteristics are defined. These include peak-to-valley value, median roughness, average roughness, and more. All of these values describe a surface vertically but hold no information about the number of peaks, distribution of peaks, and more, which is too little for the application in micro-manufacturing. The Abbott-Firestone curve, also referred to as the bearing area curve may be used for a more precise classification.

To compare two surfaces, they are expressed in their corresponding Abbott-Firestone curves, g, f. Since this type of curve represents a cumulative density function, we may use the Pearson coefficient to describe the similarity between the two surfaces, as done by [83]. In our work, we use the ideas and methods derived in [83] but compare the two curves using the L_p-norm instead.

5.3.5 Wear of Cutting Tool

During the manufacturing process, the cutting tool itself is worn down. This wear results in a change of the cutting tool shape, as can be seen in figure 5.3. The figure shows a ball-end cutting tool after using it at a specific time. Without wear, the front view would show a circle segment. The white spots mark the wear in the top view (bottom two pictures). Due to the forces acting on the cutting tool during the manufacturing process, the cutting tool starts flatting out. A simple way to model this behavior is to flatten the tool shape over time. Let the cutting edge be described via

$$\tilde{f}_t = \max\{f_t,\ c(T)\},$$

with $c(T) > 0$ a positive value that depends on the time $T \in \mathbb{R}_+$, which describes the intensity of the wear at the time T. A more accurate way would be to consider the actual forces in each point and then model material removal and deformation of the cutting tool. However, this is very costly in the sense of computational effort. In order to keep the model at an affordable complexity and enough accuracy, we use the simpler model in this work. First, we describe the degree of wear in percentage, i.e., the reduction of the minimal radius of the ball shape. Figure 5.9 provides an example. In application, wear of more than 8% is highly unlikely since the cutting tool will not function anymore. Nevertheless, for demonstration reasons, it is still plotted in figure 2 to provide an impression of the influence of wear on the tool.

Since the wear influences the cutting properties and thus the resulting surface, the cutting tool is replaced after a certain distance. The maximum cutting distance depends on the maximum wear, used workpiece material, and cutting tool material. Thus $c(T)$ is bounded above by a $c_{max} > 0$. In addition, there is no negative wear. As a result, $c(T)$ must be monotone in T.

The overall cutting distance is much larger than the cutting distance needed for one workpiece. Therefore, as a simplification, it can be assumed that the wear of the cutting tool is constant on small enough workpieces. While this simplification is not necessary, it reduces the computation times, and the error is negligible in practice. In this work, we will therefore assume that the wear of the tool at $f_0 := \tilde{f}_0$ is constant for the rest of the workpiece. Note that this is only applied to a single workpiece. We apply this simplification to each workpiece for multiple workpieces, i.e., the wear is only constant for a workpiece but not between two workpieces. Figure 5.9 is an example simulation with and without wear of the cutting tool. For this example $a_e = 1900$ and $v_f = 0.0238$ are chosen.

The mathematical operator is extended with the wear model on the cutting tool and takes an additional input t. This additional input defines how long the cutting tool has been used prior and thus provides the wear on the tool at the beginning of the simulated process.

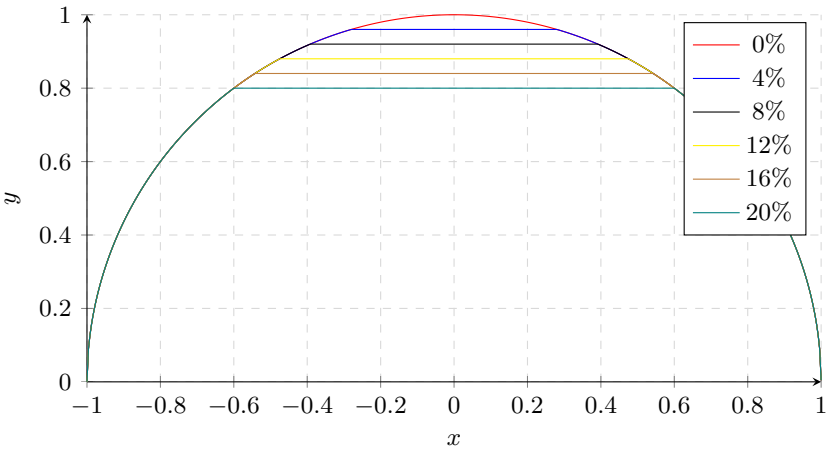

Figure 5.8: Simulation of wear on a ball-end tool.

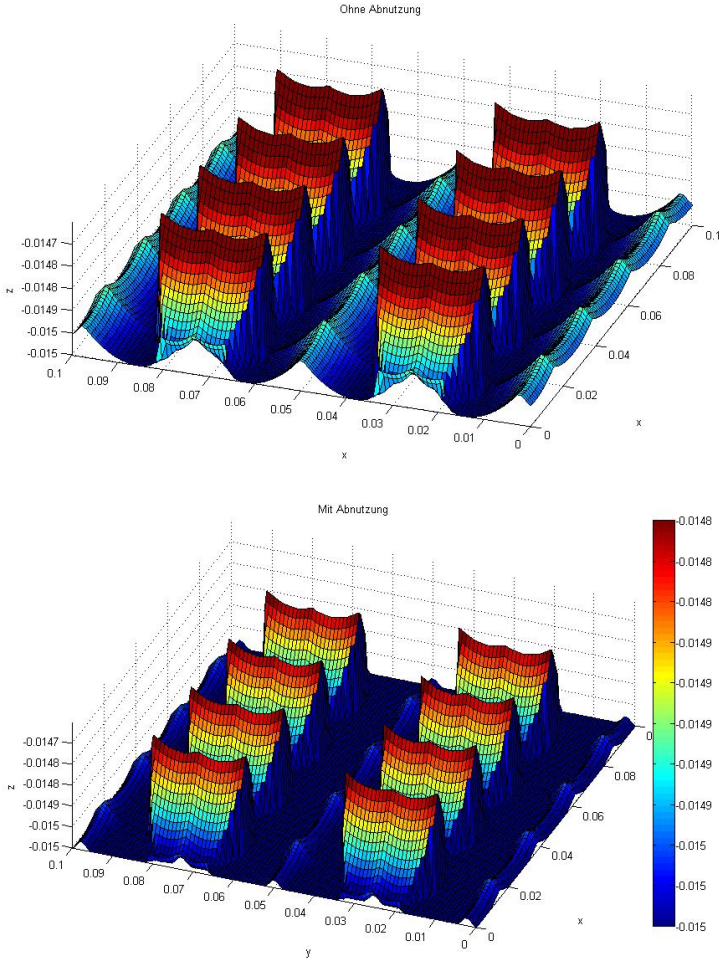

Figure 5.9: **Example simulation with wear on tool** with $v_f = 1900$ and $a_e = 0.0238$ and a wear of $c(T) = 0.01$, using the maximum height wear model.

5.3.6 Minimize deviation over fixed time interval

For a fixed time-interval, $t \in [0,\, c]$, we calculate the overall deviation from the desired tolerance via

$$\int_0^c \left\| F(p,t) - g^\delta \right\|_{L_2(\Omega),\varepsilon} dt\,.$$

This term measures how much the output will deviate over the whole time interval but does not hold any information at which time the output will not lie within the given tolerance. It, therefore, provides a soft boundary in the sense that a local minimum unequal to zero might have no point in time where

$$\left\| F(p,t) - g^\delta \right\|_{L_2(\Omega),\,\varepsilon} = 0$$

holds.

The benefit of minimizing the deviation over a fixed time interval is that we can directly apply the deviation as a discrepancy term in the Tikhonov functional

$$J^{p,q}_{\delta,\alpha,\varepsilon}(u) = \frac{1}{p} \left\| F(u) - v^\delta \right\|^p_{L_p,\varepsilon} + \alpha \mathcal{R}_q(u). \tag{5.4}$$

In addition, we can approximate the integral via known rules such as the Simpson rule.

5.3.7 Maximize time within tolerance

To maximize the period in which we can use a tool before we need to replaced it or new process parameters need to be determined, the ending time of the time-interval has to be flexible. The ending time c is then calculated through

$$c = \operatorname{argmax} \left\{ c \;\middle|\; \int_0^c \left\| F(p,t) - g^\delta \right\|_{L_2(\Omega),\varepsilon} dt = 0 \right\}.$$

If there is no $c \geq 0$ that fulfills the condition, we set $c = 0$. This choice provides the maximum time before the simulated output is outside the desired tolerance for the first time. If the discrepancy term for a parameter is greater than zero in the ideal case, the maximum for c is 0. Thus we can use argmax to replace the discrepancy term in (5.4) as was the case for fixed time-interval because parameter sets that minimize only the penalty term might be favored over terms that minimize discrepancy and penalty term.

This approach results in a min-max problem where we minimize the discrepancy and penalty while maximizing the time interval. While more accurate for the application, this approach also yields higher computational effort due to finding the argmax for each functional evaluation.

5.3.8 Numerical solution

Numerical methods are often used for parameter identification, especially for complex models. Various methods exist and can be used depending on the

properties of the mathematical operator representing the model. Boyle et al. provide an overview of existing methods in [11].

In our work the operator F is non-linear, and thus, iterative methods have to be applied. The Tikhonov functional and the maximum time-interval length have to be evaluated in each iteration. In addition, the evaluation of the operator F is computationally expensive.

One solution for these problems is to apply a two-term strategy to solve the min-max problem. First, we minimize the Tikhonov functional with incorporated tolerances to find a starting point for the min-max problem within the given tolerance. Then, the min-max problem is solved using this starting point by maximizing the time interval and minimizing the penalty term to keep regularization. We use the former discrepancy term as a non-linear boundary condition on the feasible set. This approach ensures that we fulfill all requirements of the inverse problem while optimizing the process parameter further. It also reduces the number of evaluations of the maximum time-interval length and reduces the computational effort.

For each of the two steps, we apply a different solver. For example, to solve the initial inverse problem, we use the REGINN algorithm with a conjugate gradient method [49]. This algorithm linearizes the operator and uses the conjugate gradient algorithm to solve the linear problem. We then start a new iteration with the found solution at the point where the operator is linearized.

Suppose the algorithm can find a parameter set that produces a simulated surface with the given tolerance to the provided data. In that case, we can perform the non-linear optimization for maximizing the end-time as described in section C. However, this step cannot be solved by using the REGINN algorithm as the full min-max problem does not fulfill all requirements, and a gradient method is used instead. This method is slower but has fewer conditions on the mathematical operator, which allows its application.

Overall, the two-step approach reduces the amount of performed operator evaluation by utilizing a fast converging algorithm to solve the non-linear inverse problems without wear on the tool, providing a good start estimation for the more complex min-max problem at hand.

5.4 Results

In this section, we numerically perform a parameter identification with tolerances and wear on a cutting tool for a micro-milling process. The process parameters to be identified are feed velocity v_f and radial cutting depth a_e. In addition, we fix the other process parameters to a given value.

The desired outcome is a horizontal surface with an Abbott curve g^δ, with the noise level δ. For the numerical experiment, this data is created artificially from a simulation with added noise to control the noise level directly. We choose the noise additive and normal distributed with a cutoff at δ, which we set as three times the standard deviation in the noise. The true solution is

Description	Value	Unit
Tool radius	1	mm
Tool length	45	mm
Rotation of tool	40000	rpm
Cutting depth	0.015	mm

Table 5.3: Applied process parameters

set as $(v_f,\, a_e) = (1744\frac{\text{mm}}{min},\, 0.044\text{mm})$ and the tolerance $\varepsilon = 25^{-6}\text{mm}$. Further process parameters that we do not need to identify are in table 5.3.

Before starting the parameter identification, we define the type of regularization and a starting point for the numerical algorithm. We choose $p^0 = (2000,\, 0.02)$ for the starting point. This starting point is the suggested parameter set found for the process by computer-aided design (CAD) without considering surface structures. The starting point for the numerical algorithm is set to $p^0 = (2000,\, 0.01)$ which produces a surface with an Abbott curve outside the given tolerance.

To compare the solution of the min-max problem, we perform a classic parameter identification without considering tolerances and wear. In this case the parameter $p = (174448,\, 0.0442)$ is found. Figure 5.10 top shows the simulated surface of the found solution. While it is a good approximation for the true solution, it will be outside the accepted area with wear of about 0.75%. The solution resides only in the lower half of the accepted area when wear occurs in the figure.

The solution of the entire min-max problem is $p = (1784,\, 0.04518)$ which is further away from the true solution but is still inside the given tolerance. Figure 5.10 bottom visualizes the solution and shows that the output stays within the accepted area for a longer time if wear occurs. In this case, wear of 0.8% is still within the accepted area. Compared to the former, the new solution uses the whole given range of tolerance.

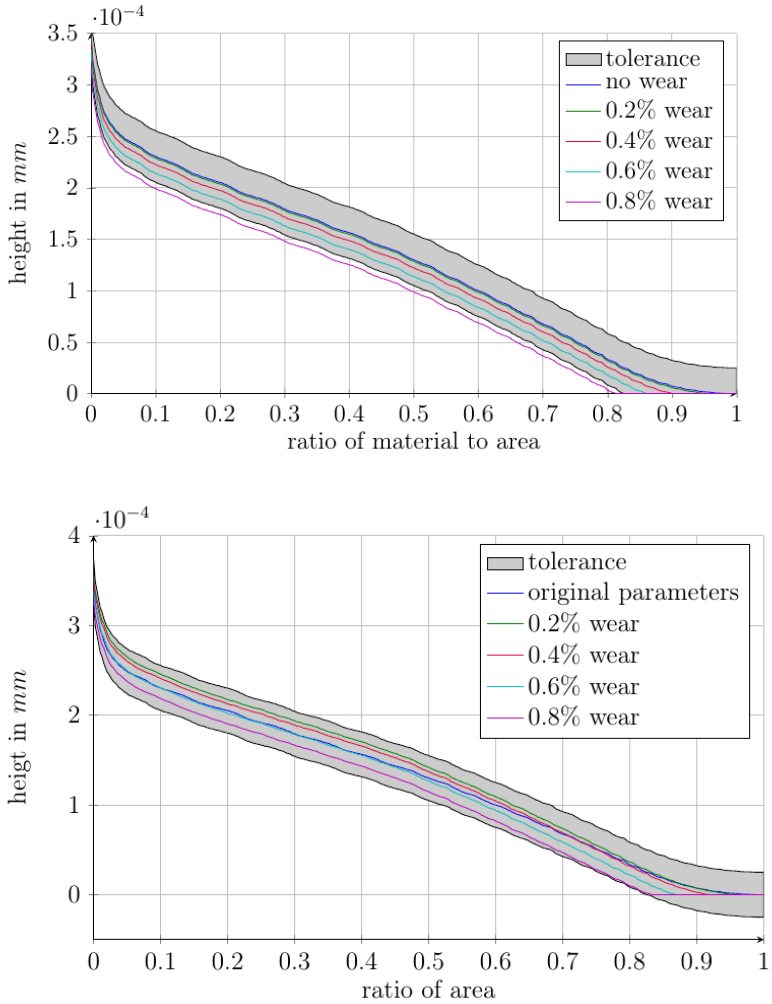

Figure 5.10: Solution of parameter identification considering wear of tool. **Top:** Solution for Tikhonov functional without tolerance and no optimization considering wear of cutting tool.
Bottom: Solution for Tikhonov functional with tolerance and additional optimization considering wear of cutting tool.

6 Conclusion

This work introduces inverse problems with regularization by incorporating a tolerances in the discrepancy term of Tikhonov type functionals. We use the hinge loss, that is commonly used for support vector regression and is defined as

$$|\lambda|_\varepsilon := \begin{cases} 0 & |\lambda| \leq \varepsilon \\ |\lambda| - \varepsilon & \text{o.w.} \end{cases} \tag{6.1}$$

to model the tolerance. This loss function is extended for functions $f \in L_p(\Omega)$ via

$$\|f\|^p_{L_{p,\varepsilon(x)}} = \int_\Omega |f(x)|^p_{\varepsilon(x)} \, \mathrm{d}x, \tag{6.2}$$

where $\varepsilon(x) : \Omega \to \mathbb{R}_0^+$ and $\varepsilon(x) < c < \infty$ for all $x \in \Omega$. This point-wise ε −insensitive loss function accurately models confidence bands and offers an extra regularization. We provide examples which illustrate that the tolerances in the discrepancy term leads to a more stable reconstruction compared to the classical Tikhonov-type functionals.

This thesis established the mathematical foundation for Tikhonov-type functions with an ε-insensitive discrepancy term. This foundation includes proof of the existence of a minimizer, stability, and convergence. We compare our approach to existing methods and show how it differs from them.

In order to apply the theory in practice, we need a method to minimize the given functional. We explored existing numerical methods to find a minimizer of the altered Tikhonov-type functional and compared them to our new introduced subgradient method with adaptive step-size. The focus are numerical methods that can minimize non-smooth functionals. Numerical solvers for non-smooth functions can minimize

$$J^{p,q}_{\delta,\alpha,\varepsilon}(u) = \frac{1}{p} \left\| F(u) - v^\delta \right\|^p_{L_{p,\varepsilon}} + \alpha \mathcal{R}_q(u) \tag{6.3}$$

for $p, q \leq 1$ and regularization terms. In addition, we derive how existing solvers for support vector machines can be applied on discretized inverse problems with linear forward operators, $p = \{1, 2\}$, and $\mathcal{R}_q(u) = \|u\|_q$ with $q = 2$.

The altered Tikhonov-type functionals can be used in various applications such as image denoising, one-dimensional signal denoising, and better capture tolerance or multi-measurements. We provide multiple examples for each of these application fields. The primary real-world application is micro-milling. We explain the micro-milling process model and extend an existing method to capture wear on the cutting tool. This extended method is used for parameter identification with tolerances by solving a min-max problem.

Overall we provide the full picture of our proposed method, starting from the mathematical foundation, continuing to numerical solutions, and ending with example applications.

For future research on this topic, we would focus on further generalizing the presented theory and exploring the connection between ε-insensitive loss and sparsity in the reconstruction. In addition, the evaluation of different non-smooth solvers that led to the development of our own step-size adaptive subgradient method revealed that further improvements in this area are possible. We will investigate adaptive momentum for subgradients methods in the future.

Bibliography

[1] *CNC-Handbuch 2009/2010: CNC, DNC, CAD, CAM, FFS, SPS, RPD, LAN, CNC-Maschinen, CNC-Roboter, Antriebe, Simulation, Fachwortverzeichnis.* Hanser, München, 2009. 551 S. : Ill., graph. Darst.

[2] Werner Alt. *Nichtlineare Optimierung - Eine Einführung in Theorie, Verfahren und Anwendungen.* Vieweg+Teubner / Vieweg+Teubner Verlag, 2011.

[3] Yusuf Altintas. *Manufacturing Automation - Metal CUtting Mechanics, Machine Tool Variations, and CNC Design.* Cambridge University Press, 2006.

[4] A. Amacheron and P.T. Mativenga. Size effect and tool geoemetry in micromilling of tool steel. *Precision Engineering*, 2009.

[5] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, Inc., 1996.

[6] W.Y. Bao and I.N. Tansel. Modeling micro-end-milling operations. part i: analytical cutting force model. *International Journal of Machine Tools and Manufacture*, 40(15):2155–2173, 2000.

[7] Heinz H. Bauschke and Patrick L. Combettes. Convex analysis and monotone operator theory in hilbert spaces. In *CMS Books in Mathematics*, 2011.

[8] Antonio Bolufé-Röhler and Stephen Chen. Multi-swarm hybrid for multimodal optimization. pages 1759–1766, 07 2012.

[9] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1899–1903, 2017.

[10] S. Boyd, J. Duchi, and L. Vandenberghe. Subgradients - lecturenote. Stanfdord University, online, 2018.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] M. Burger and S. Osher. Convergence rates of convex variational regularization. *Inverse Problems*, 2004.

[13] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence. 8(6):679–698*, 1986.

[14] George Casella and Roger L. Berger. *Statistical Inference*. China Machine Press, 2nd edition, 2012.

[15] K. Chaitanya, D. Somayajulu, and P. R. Krishna. A pso based community detection in social networks with node attributes. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018.

[16] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] Changyou Chen, David Carlson, Zhe Gan, Chunyuan Li, and Lawrence Carin. Bridging the gap between stochastic gradient mcmc and stochastic optimization. 12 2015.

[18] Chih-Jen Lin Chia-Hua Ho. Large-scale linear support vector regression. *Journal of Machine Learning Research*, 2012.

[19] R. D. da Silva, R. Minetto, W. R. Schwart, and H. Pedrini. Adaptive edge-preserving image denoising using wavelet transforms. *Springer Journal,*, pages 567–580, 2013.

[20] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[21] U. P. Diliman E. P. Adorio. Mvf - multivariant test functions library in c for unconstrained global optimization. http://http://www.geocities.ws/eadorio/mvf.pdf, 2005.

[22] A. E. Eiben, E. H. L. Aarts, and K. M. Van Hee. Global convergence of genetic algorithms: A markov chain analysis. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 3–12, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[23] O. Riemer F. Böhmermann, W. Preuß. Manufacture and functional testing of micro forming tools with well-defined tribological properties. *Proceedings of the 29th ASPE Annual Meeting*, page 486–491, 2015.

[24] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[25] Fuchang Gao and Lixing Han. Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51:259–277, 2012.

[26] Juan Manuel Górriz, Carlos G. Puntonet, Moisés Salmerón, and Fernando Rojas Ruiz. Hybridizing genetic algorithms with ica in higher dimension. In Carlos G. Puntonet and Alberto Prieto, editors, *Independent Component Analysis and Blind Signal Separation*, pages 414–421, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[27] Phil Gralla, Iwona Piotrowska-Kurczewski, and Peter Maaß. Tikhonov functionals incorporating tolerances. *PAMM*, 17(1):703–704, 2017.

[28] Markus Grasmair, Markus Haltmeier, and Otmar Scherzer. Sparse regularization with l^q penalty term. *Inverse Problems*, 24:055020 (13pp), 2008.

[29] P. Gupta and A. Garg. Image denoising using bayesshrink method based on wavelet transform. *International Journal of Electronic and Electrical Engineering*, 8(1):33–40, 2015.

[30] Lixing Han and Michael M. Neumann. Effect of dimensionality on the nelder–mead simplex method. *Optimization Methods and Software*, 21:1 – 16, 2006.

[31] A. Neubauer Heinz Werner Engl, Martin Hanke. *Regularization of Inverse Problems*. Springer Dordrecht, 2000.

[32] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *arXiv preprint arXiv:2106.15282*, 2021.

[33] B Hofmann, B Kaltenbacher, C Pöschl, and O Scherzer. A convergence rates result for tikhonov regularization in banach spaces with non-smooth operators. *Inverse Problems*, 23(3):987–1010, apr 2007.

[34] I. Piotrowska-Kurczewski, J. Schlasche, D. Weimer, B. Scholz-Reiter, and P. Maass. Image denoising and quality inspection of micro components using perona-malik diffusion. *Procedia CIRP,*, 8:432–437, 2013.

[35] Stemmer Imaging. *Das Handbuch der Bildverarbeitung*. Stemmer Imaging, 2016.

[36] Bangti Jin and Peter Maass. Sparsity regularization for parameter identification problems. *Inverse Problems*, 28(12):123001, 2012.

[37] Stephen Wright Jorge Nocedal. *Numerical Optimization*. Springer, New York, NY, 1999.

[38] S. Kabanikhin, N Tikhonov, V Ivanov, and M Lavrentiev. Definitions and examples of inverse and ill-posed problems. *Journal of Inverse and Ill-posed Problems - J INVERSE ILL-POSED PROBL*, 16:317–357, 01 2008.

[39] Barbara Kaltenbacher, Frank Schöpfer, and Thomas Schuster. Iterative methods for nonlinear ill-posed problems in banach spaces: convergence and applications to parameter identification problems. *Inverse Problems*, 25(6):065003, apr 2009.

[40] K S Kazimierski, P Maass, and R Strehlow. Norm sensitivity of sparsity regularization with respect to to p. *Inverse Problems*, 28(10):104009, oct 2012.

[41] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995.

[42] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *CoRR*, abs/2004.11362, 2020.

[43] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[44] Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[45] Krzysztof C. Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical Programming*, 2001.

[46] T. Kolda, R. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.*, 45:385–482, 2003.

[47] Yury Korolev and Jan Lellmann. Image reconstruction with imperfect forward models and applications in deblurring. *arXiv:1708.01244v3 [cs.NA] 23 Oct 2017*, 2017.

[48] Vipul Kothari, J. Anuradha, Shreyak Shah, and Prerit Mittal. *A Survey on Particle Swarm Optimization in Feature Selection*, volume 270, pages 192–201. 01 2012.

[49] Armin Lechleiter and Andreas Rieder. Towards a general convergence theory for inexact newton regularizations. *Numerische Mathematik*, 114, 01 2010.

[50] Prof. Armin Lechleiter. *Inverse Probleme 1.* Fachbereich 3, Universität Bremen, 2012. Skript zur Vorlesung.

[51] Shih-Wei Lin, Zne-Jung Lee, Shih-Chieh Chen, and Tsung-Yuan Tseng. Parameter determination of support vector machine and feature selection using simulated annealing approach. *Applied Soft Computing*, 8(4):1505 – 1512, 2008. Soft Computing for Dynamic Data Mining.

[52] Dirk Lorenz and Nadja Worliczek. Necessary conditions for variational regularization schemes. *Inverse Problems*, 2013.

[53] Alfred K. Louis. *Inverse und schlecht gestellte Probleme.* Teubner, Stuttgard, 2001.

[54] Kilmer M. Belge, M. E. and E. Miller. Efficient determination of multiple regularization parameters in a general l-curve framework. *Inverse Problems 18*, 1999.

[55] C. Smutnicki M. Molga. Test functions for optimization needs. http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf, 2005.

[56] K. I. M. McKinnon. Convergence of the nelder–mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158, 1998.

[57] Vladimir Alekseevich Morozov. *Methods for solving incorrectly posed problems.* Springer Science & Business Media, 2012.

[58] K. Cheng N. F. M. Aris. Characterization of the surface functionality on precision machined engineering surfaces. *The International Journal of Advanced Manufacturing Technology*, 2008.

[59] Mathew M. Noel. A new gradient based particle swarm optimization algorithm for accurate computation of global minimum. *Applied Soft Computing*, 12(1):353–359, 2012.

[60] Iwona Piotrowska Kurczewski and Jost Vehmeyer. Simulation model for micro-milling operations and surface generation. In *Modelling of Machining Operations*, volume 223 of *Advanced Materials Research*, pages 849–858. Trans Tech Publications Ltd, 6 2011.

[61] Mangal Prakash, Alexander Krull, and Florian Jug. Fully unsupervised diversity denoising with convolutional variational autoencoders. In *International Conference on Learning Representations*, 2021.

[62] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[63] Andreas Rieder. *Keine Probleme mit Inversen Problemen. Eine Einführung in ihre stabile Lösung.* Vieweg & Sohn Verlag, 2003.

[64] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv:2104.07636*, 2021.

[65] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[66] Javad Salimi Sartakhti, Homayun Afrabandpey, and Mohamad Saraee. Simulated annealing least squares twin support vector machine (salstsvm) for pattern classification. *Soft Computing*, 21(15):4361–4373, Aug 2017.

[67] Javad Salimi Sartakhti, Mohammad Hossein Zangooei, and Kourosh Mozafari. Hepatitis disease diagnosis using a novel hybrid method based on support vector machine and simulated annealing (svm-sa). *Computer Methods and Programs in Biomedicine*, 108(2):570 – 579, 2012.

[68] Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, and Frank Lenzen. *Variational Methods in Imaging.* Springer, 2009.

[69] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels - Support Vector Machines, Regularization, Optimization, and Beyond.* The MIT Press, 2002.

[70] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 5th edition, 2015.

[71] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

[72] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, pages 199–222, 2014.

[73] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959. Curran Associates, Inc., 2012.

[74] Kenneth O. Stanley and Jason Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, page 2009, 2009.

[75] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[76] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[77] Robert Tibshirani Trevor Hastie and Jerome Firedman. *The Elements of Statisrical Leraning - Datamining, INference, and Predicition*. Springer, 2016.

[78] Gabriel Turinici. The convergence of the stochastic gradient descent (sgd) : a self-contained proof. Technical report, 2021.

[79] H. K. Tönshoff and B. Denkena. *Spanen - Grundlagen*. Springer, 2004.

[80] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. *CoRR*, abs/1711.10925, 2017.

[81] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, NY, 1995.

[82] Jost Vehmeyer. *Geometrische Modellierung und funktionsbezogene Optimierung der inhärenten Textur von Mikrofräsprozessen*. PhD thesis, University Bremen, 2016.

[83] Jost Vehmeyer, Iwona Piotrowska-Kurczewski, Florian Böhmermann, and Peter Maaß. Leas-square based parameter identification for a function-related surface optimisation in micro ball-end milliing. In *15th CIRP Conference on Modelling of Mchining Operations*, 2015.

[84] Jost Vehmeyer, Iwona Piotrowska-Kurczewski, and Sven Twardy. A surface generation model for micro cutting processes with geometrically definied cutting edges. In *37th Matador Conference*. 2013.

[85] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef

Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[86] F. Vollertsen. Categories of size effects. *Production Engineering*, 4(2):377–383, 2008.

[87] Jacques Wainer and Pablo Fonseca. How to tune the rbf svm hyperparameters: An empirical evaluation of 18 search algorithms, 2020.

[88] Xin-She Yang. Chapter 8 - particle swarm optimization. In Xin-She Yang, editor, *Nature-Inspired Optimization Algorithms (Second Edition)*, pages 111–121. Academic Press, second edition edition, 2021.

[89] Xin-She Yang, Suash Deb, and Simon Fong. Accelerated particle swarm optimization and support vector machine for business optimization and applications. In Simon Fong, editor, *Networked Digital Technologies*, pages 53–66, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[90] E. Zeidler. *Nonlinear functional analysis and its application III: Variational Methods and Optimization*. Springer, New York, NY [u.a.], 1985.

[91] Q. Zhang, G. Shan, X. Duan, and Z. Zhang. Parameters optimization of support vector machine based on simulated annealing and genetic algorithm. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1302–1306, Dec 2009.

[92] Y. Zhou, Z. Li, H. Zhou, and R. Li. The application of pso in the power grid: A review. In *2016 35th Chinese Control Conference (CCC)*, 2016.

Eidesstattliche Erklärung

Ich versichere die vorliegende Dissertation ohne fremde Hilfe angefertigt zu haben. Ich habe keine andere als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.