Universität Hamburg

Master's Thesis

# Learned operator correction for the proximal gradient method

by

*Björn Ehlers*
Matrikel-Nr. 6950454

supervisor and first examiner:
PD Dr. Tobias Kluth

second examiner:
Prof. Dr. Armin Iske

Date of submission: 21 September 2022

# Learned operator correction for the proximal gradient method

Björn Ehlers

Universität Hamburg, Fachbereich Mathematik

## Contents



**Abstract**

In this master's thesis I followed the idea, of a learned operator correction approach to deal with imprecise operators in linear inverse problems, given by Lunz et al. in [15]. I derived a theoretical result for the convergence of the proximal descent method, using the corrected operator, to a neighbourhood, of a minimizer of the objective function,

obtained with the precise operator. This Theory is similar to what is done by Lunz et al. for the gradient descent method, with the advantage that the objective function does not need to be differentiable. In addition to the theory I implemented the training of corrections for the Radon transform in case of an imprecise operator inspired by Nano-CT,i.e. an impreciseness caused by an unknown shifting of the phantom during the measurement. Using analytic phantoms the corrected operator is, for shifts similar to the ones the correction is trained on, a small improvement over the uncorrected one.

# 1 Introduction

In inverse problems we often want to solve

$$Ax = y$$

with a linear operator $A : X \to Y$. There are many approaches on how to deal with noise in the data $y$. One a approach is to minimize an objective function $\mathcal{L} : X \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$

$$\min_{x \in X} \mathcal{L}(x) = \min_{x \in X} \|Ax - y\|^2 + \lambda \Lambda(x), \tag{1.1}$$

with a regularization term $\lambda\Lambda(x)$.

A different problem arises when we only have an imprecise operator $\tilde{A}$ instead of $A$. Merely solving

$$\min_{x \in X} \tilde{\mathcal{L}}(x) = \min_{x \in X} \left\| \tilde{A}x - y \right\|^2 + \lambda \Lambda(x) \tag{1.2}$$

is not useful, because it leads to a large error in the reconstruction.

One example where this problem arises is in computer tomography (CT). When we know the exact measurement procedure, i.e. the exact positioning of the measured object and the positioning of the X-ray beams used to measure it, we can model $A$ as the Radon transform and can reconstruct our object. Unfortunately, we do not have the exact positing of the object and the beams. While the positioning of the beams is precise enough, we do not know the internal movement of the object during the measurement. Such an internally moving object can be for example a beating heart.

Even when we are not looking at an object that is moving internally we can still run into problems. When the resolution of the CT lies in the realm of nanometres, we refer to this as nano CT, see [11]. The problem here is that in order to take measurements in CT, we need to spin either the source and detector of the X-ray beams or the object. This can lead to small wobbles of the measurement architecture and due to the high resolution, even small deviations from the assumed paths of the X-ray beams in respect to the object can lead to problems in the reconstruction for nano CT.

One approach is to try to measure the movement that occurs during the measurement process and correct the operator accordingly, or to measure indicators of the movement (e.g. heartbeat in medical CT and then model the resulting movement and correct for it, see for example [19, 8, 22]). This involves more effort and is not always feasible.

One other approach is given by the authors of the paper "On learned operator correction in inverse problems" [15]. Their idea is to use machine learning models as corrections for the imprecise $\tilde{A}$, with the knowledge of $A$, and get a corrected operator

$$\min x \in X \mathcal{L}_\Theta(x) = \min x \in X \, \|A_\Theta x - y\| + \lambda\Lambda(x). \tag{1.3}$$

and a correction for the adjoint $A^*_\Phi$ of the operator. The corrected operators are then used in the gradient descent method to solve the minimization problem. Lunz et al. [15] also show that if the corrected operators are close enough to the precise operator, we can get close to the solution we would get if we had this precise operator $A$.

To be able to use the gradient descent method, we need the objective function $\mathcal{L}$ to be differentiable, i.e. the regularization term $\lambda\Lambda$ needs to be differentiable. We would like to use a non-differentiable regularization term like the $L^1$-norm. In this Master's thesis, I take the idea of [15] and demonstrate a similar theory for the proximal gradient method, which has the advantage that $\lambda\Lambda$ only has to be proper, lower semi continuous and convex.

I also show that we can relax the additional assumption of strong convexity of the objective function $\mathcal{L}$, which is needed in [15], to $\mathcal{L}$ being a convex lower semi-continuous function and the minimizers of $\mathcal{L}$ forming a closed set.

In addition to that I report how I implemented the training of corrections for the Radon transform. The impreciseness of the used operators is inspired by the problem of nano CT, i.e. wobbles causes by shifts of the phantom 's position, without deformations of the phantom. For the implementation I used only analytic ellipse phantoms.

I experimented with different positioning of the U-nets that are used for the corrections and was able to make small improvement with the learned correction operator over the imprecise operator. This works also on operators the model was not trained on, as long as they are similar enough.

In the following sections, we will first look at some background theory of computer tomography, inverse problems, convex optimization, and machine learning, followed by the theory from the aforementioned paper [15]. After that we will look at the new theory for the correction in the proximal gradient method, followed by information about the implementation and the setup of the numerical experiments. We will then look at the result s of these experiments and will then conclude with the findings of this Master's thesis.

# 2 Background

## 2.1 Computer tomography (CT)

First we get an introduction to computer tomography (CT) based on [3, 20]. The idea is to make use of the fact that different materials or different densities of the material absorb X-rays to different amounts. When we shoot an X-ray beam trough an object, it gets absorbed or scattered depending on the composition of the material. The physics and the technical side of CT are discussed in [7].

We can model the object we want to reconstruct as an *attenuation function* $f : \mathbb{R}^N \to \mathbb{R}$ that describes the proportion of X-ray radiation that is absorbed from the object. $N$ can be either 2 or 3. In this thesis we will only look at 2D objects, so for us $N$ will be 2. To construct a mathematical model, we assume that the X-rays are only being absorbed and not scattered, the beams have zero width and all X-rays in the beam have the same energy. The last assumption is necessary because the absorption depends on the energy of the X-rays. When we look at the change of the intensity $I(x)$ of the X-ray beam, we get

$$\frac{\mathrm{d}}{\mathrm{d}x} I(x) = -f(x)I(x).$$

Taking the integral over the line $\ell$ between the source of the X-rays $x_S$ and the point on the detector $x_D$ gives us

$$\int_{I_S}^{I_D} \frac{1}{I} \mathrm{d}I = \int_{\ell} f(x) \mathrm{d}x,$$

with $I_S$ being the beam intensity at the source and $I_D$ the intensity measured at the detector. We can compute the integral on the left-hand side and get

$$\log\left(\frac{I_S}{I_D}\right) = \int_{\ell} f(x) \mathrm{d}x. \tag{2.1}$$

For the purposes of this thesis, equation (2.1) is enough and we can compute the left-hand side with what we get from our measurements.

We can characterize a line in $\mathbb{R}^2$ by an angle $\varphi \in [0, 2\pi[$ and a length $s \in \mathbb{R}_{\geq 0}$

$$\ell_{\varphi,s} := \left\{ x \in \mathbb{R}^2 \middle| \langle x, n(\varphi) \rangle = s \right\}.$$

**Definition 2.1** (Radon transform)**.** We call

$$\mathcal{R}f(s, \varphi) = \int_{\ell_{\varphi,s}} f(x) \mathrm{d}x$$

the *Radon* or *ray transform*.

There are now two ways to look at the problem of reconstruction. On the one hand we could try to solve the problem analytically first, with all possible lines and accurate data, and then discretize and deal with ill-posedness of the reconstruction problem. This approach would give us the so-called filtered-back projection discussed in more detail in [18]. On the other hand, we can first discretize our problem and get a matrix equation to solve which would still be somewhat ill-conditioned.

In this thesis we will use only algebraic reconstruction techniques, i.e., we discretize our problem before we try to solve it. To make a reconstruction of $f$, we need measured data from multiple beams. The way the beams are set up is called *measurement geometry*. In this thesis we use the fan beam geometry, see Figure 1, where we have one source and opposite to that a detector with multiple detection points. Together they either rotate around the object, or the object is rotated and measurements are taken at uniformly spaced angles.

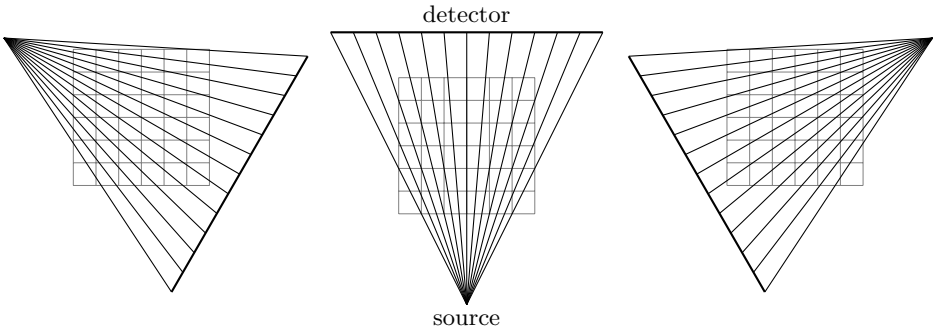



Figure 1: Fan beam geometry

We discretize our problem by defining a pixel grid that consists of $n$ by $m$ pixels that are all squares of the same size and assuming our phantom is constant on the pixels The integral from equation (2.1) then simplifies to

$$\int_\ell f(x)\mathrm{d}x = \sum_{i=1}^{n}\sum_{j=1}^{m} f_{i,j} \cdot l_{i,j}(\ell)\mathrm{d}x,$$

where $f_{i,j}$ is the value of $f$ on the pixel $p_{i,j}$ and $l_{i,j}(\ell)$ the length of the intersection of $\ell$ with $p_{i,j}$.

The linear ray transform for the multiple beams can be combined into one operator which we will call *ray transform*. It can also be expressed as a matrix called the *Radon matrix*.

We will denote this combined operator as $\mathcal{R} : X \to Y$, with the pixel or phantom space $X = \mathbb{R}^N$ for phantoms with $N$ pixels and the measurement space $Y = \mathbb{R}^M$ for $M$ measurements. The measurement data in CT is also called a *sinogram*.

Now we can formulate our reconstruction problem as solving

$$\mathcal{R}x = y,$$

for $x$. This type of problem is called an *inverse problem*.

## 2.2 Inverse problems

In general, if we have two normed spaces $X, Y$, an operator $A : X \to Y$ and an element $y \in Y$, we call

$$A(x) = y \tag{2.2}$$

an *inverse problem*, the following is based on [4, 20]. In our case, the ray transform is linear and the $\mathbb{R}^N$ are Hilbert spaces, therefore we will focus here on linear $A \in \mathscr{L}(X, Y)$ with $X, Y$ Hilbert spaces.

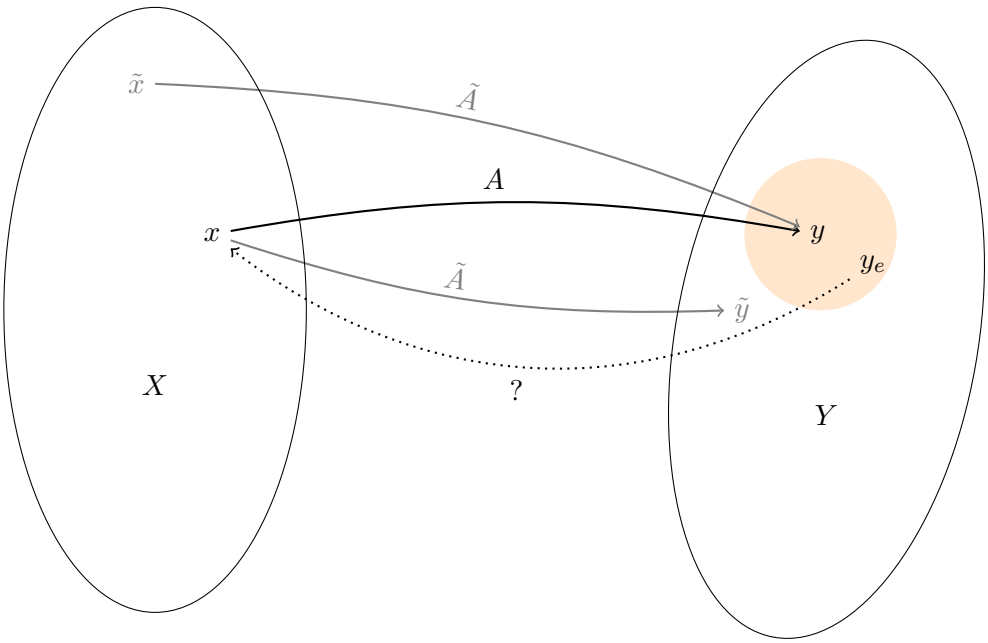



Figure 2: Inverse problems illustration
We have the problem that we know $A$ and $y$ and want to get back to $x$. In addition we only know a point $y_e$ in the neighbourhood of $y$ and in our case not even $A$ and just te in correct $\tilde{A}$.

As shown in Figure 2, we would like to get an inverse operator of $A$ which is also able to handle noise in $Y$. Unfortunately, there are three things that can go wrong, if one of these does, we call our problem *ill-posed*.

**Definition 2.2** (ill-posedness)**.** Let $X, Y$ be normed spaces and $A : X \to Y$ a mapping. Then the problem $(A, X, Y)$ is *well-posed* if

1. we have at least one solution $x \in X$ for all $y \in Y$ such that $A(x) = y$,
2. the solution $x$ is unique, and
3. the inverse $A^{-1} : Y \to X$ is continuous, i.e. $x$ depends continuously on $y$.

If these conditions do not apply, the inverse problem is called *ill-posed*.

In the case of the Radon matrix we can see that we might have trouble to fulfil conditions 1 and 2 of well-posedness.

1. We do not have a solution for every $y$ if we have an overdetermined system of equations, which we get if we have more measurements than pixels.
2. Our solution is not unique if we have an underdetermined system of equations, which we get if we have fewer measurements than pixels.

To address the first point, we can solve the *least squares problem*

$$\min_{x \in X} \|Ax - y\|.$$

instead of (2.2). If we assume that $Ax - y$ has a solution, then it is also a minimizer of the least squares problem, and if we have no solution, the minimizer is at least an element of $X$ that has the smallest possible residual.

We also get an alternative approach to solving the minimization problem.

**Theorem 2.3.** *Let $A \in \mathscr{L}(X, Y)$, then the following is equivalent:*

1. *$\bar{x}$ is a minimizer of the least squares problem*

$$\min_{x \in X} \|Ax - y\|_2. \tag{2.3}$$

2. *$\bar{x}$ is the solution of the* normal equation

$$A^*Ax = A^*y. \tag{2.4}$$

*Proof.* $1 \to 2$:

Let $\bar{x}$ be a minimizer of

$$\min_{x \in X} \|Ax - y\|_2,$$

then it is also a minimizer of

$$\min_{x \in X} \frac{1}{2}\|Ax - y\|_2^2.$$

This is differentiable in $x$ with the derivative $A^*(Ax - y)$. With the first order optimality condition, we get

$$A^*(A\bar{x} - y) = 0.$$

$2 \to 1$: Let $\bar{x}$ be a solution of the normal equation, then

$$A^*(Ax - y) = 0.$$

This holds if and only if $Ax - y \in \ker(A)$ the null space of $A^*$ and $\ker(A^*) = \mathbb{R}range(A)^\perp$. Let $u \in X$, then we have to show

$$\|Au - y\|_2 \geq \|A\bar{x} - y\|_2 \quad \forall u \in X.$$

We can split

$$Au - y = \underbrace{Au - Ax}_{\in \text{Range}(A)} + \underbrace{Ax - y}_{\in \text{Range}(A)^\perp}.$$

Together with Pythagoras that gives us

$$\|Au - y\|_2^2 = \|Au - Ax\|_2^2 + \|Ax - y\|_2^2,$$

and from that we get

$$\|Au - y\|_2 \leq \|Ax - y\|_2.$$

□

Solving these problems has the advantage that a solution always exists. The problem of uniqueness still remains, but the uniqueness can be forced if we always choose the element with the smallest norm. These two approaches are combined in the generalized inverse.

**Definition 2.4** (generalized inverse)**.** We define

$$A^g : \text{Range}(A) \oplus \text{Range}(A)^\perp \subset Y \to X, \quad y \to x^g$$

where $x^g$ is the element from the solution space of the normal equation that has the smallest norm.

The third criterion for well-posedness is no problem in our finite dimensional setting. We could show that the pseudo inverse is continuous if and only if $\text{Range}(A)$ is closed (see [20]), which is always the case in finite dimensions. In infinite dimensions we would see that this last criterion is most often not fulfilled. But even in finite dimensions just using the pseudo inverse often amplifies data noise in the reconstruction, due to the structure of the operator.

To address that issue, we can add an additional term to the minimization problem

$$\min_{x \in X} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \Lambda(x). \tag{2.5}$$

We call $\lambda$ the regularization parameter and $\Lambda$ a penalizing term. A regularization is defined as follows.

**Definition 2.5** (regularization)**.** Let $A \in \mathscr{L}(X, Y)$ and $\{R_t\}_{t>0}$ be a family of continuous operators mapping from $Y$ to $X$ with $R_t(0) = 0$. If there is a mapping $\gamma : ]0, \infty[ \times Y \to ]0, \infty[$ such that for every $y \in \text{Range}(A)$

$$\sup \left\{ \left\| A^g y - R_{\gamma(\delta, y^\delta)} y^\delta \right\|_X \mid y^\delta \in Y, \left\| y - y^\delta \right\| \leq \delta \right\} \to 0 \quad \text{for } \delta \to 0,$$

then the pair $\left(\{R_t\}_{t>0}, \gamma\right)$ is called a *regularization method* for $A^g$. We call $R_t$ a linear regularization if all $R_t$ are linear. The mapping $\gamma$ is called *parameter choice rule* and is defined in a way such that

$$\lim_{\delta \to 0} \sup \left\{ \gamma\left(\delta, y^\delta\right) \mid g^\delta \in Y, \left\| y - y^\delta \right\| \leq \delta \right\} = 0.$$

The number $\gamma\left(\delta, y^\delta\right)$ is called *regularization parameter* .

This definition ensures that the smaller the error, the closer we are to the original problem, and in the limit where the error $\delta$ converges to zero, we get back the original problem. For theory on the quality of reconstructions, see [20]. In our finite dimensional case, the penalty term and regularization parameter are chosen in a heuristic approach where we can also use our prior knowledge about the solution. The goal is to choose our penalty term in a way that it is small for the solutions we expect. For example:

- $\Lambda = \|\cdot\|_2^2$ if the solution is assumed to be smooth, the resulting regularization is the so called Thikonov regularization. We choose
- $\Lambda = \|\cdot\|_1$ or $\Lambda = \|\nabla \cdot\|_1$ for solutions with sharp edges.

For the choice of the regularization parameter $\lambda$, both the noisy data $y^\delta$ and the error level $\delta$ are needed, otherwise it can be proofed that we do not get a regularization, see [20]. It is possible to choose a heuristic $\lambda$ with out The regularization parameter is chosen experimentally, see [20].

An experimental approach to find a good $\lambda$ is to compute a reconstruction for multiple different values of $\lambda$ and then use the one that gives the in some sense best reconstruction. On criteria used there is the L-curve criteria, see [20]

Some attempts have been made to train a neural network to choose the regularization parameter or even to replace the whole regularization term with a neural net [16, 14, 17].

We can look at the *total error* and split it into *approximation error* and *data error*

$$\|A^+ g - R_t g_e\| \leq \underbrace{\|A^+ g - R_t g\|}_{\text{approximation error}} + \underbrace{\|R_t(g - g_e)\|}_{\text{data error}}$$

For choosing a $\lambda$, the total error is in practice not a good choice, because the original object $p$ and the data without noise $g$ are not known.

**Example 2.6.** *We can compute that our $L^2$ norm penalty term gives us a regularization. The $\|x\|_2$ term always ensures that $x \in N(A)^\perp$ and we have $A^g y = \arg\min_{x \in N(A)^\perp} \|Ax - y\|_2$, so we get*

$$\begin{aligned}
R_\lambda(y) &= \arg\min_{x \in X} \frac{1}{2\lambda}\|Ax - y\|_2^2 + \frac{1}{2}\|x\|_2^2 \\
&= \arg\min_{x \in N(A)^\perp} \|Ax - y\|_2^2 + \lambda\|x\|_2^2 \\
&\xrightarrow{\lambda \to 0} \arg\min_{x \in N(A)^\perp} \|Ax - y\|_2^2 \\
&= \arg\min_{x \in N(A)^\perp} \|Ax - y\|_2 \\
&= A^g y.
\end{aligned}$$

### 2.2.1 Operator error

If we do not have the correct operator, $A$, but instead only an imprecise operator $\tilde{A}$ we can estimate the total error with

$$\|p - R_\lambda(\tilde{A}, g_e)\| \leq \|p - R_\lambda(A, g)\| + \|R_\lambda(A, g) - R_\lambda(\tilde{A}, g_e)\|$$

We could now treat this as if $\|R_\lambda(A, g) - R_\lambda(\tilde{A}, g_e)\|$ were just the data error and try using a regularization approach as before. The problem is, that the error is often too large to be dealt with in that way. We have a choice how we split the term into operator and data error. We can either compute the operator error at $g$ and the data error with $\tilde{A}$

$$\|R_\lambda(A, g) - R_\lambda(\tilde{A}, g_e)\| \leq \|R_\lambda(A, g) - R_\lambda(\tilde{A}, g)\| + \|R_\lambda(\tilde{A}, g) - R_\lambda(\tilde{A}, g_e)\|$$

or compute the data error with $A$ and the operator error at $g_e$

$$\|R_\lambda(A, g) - R_\lambda(\tilde{A}, g_e)\| \leq \|R_\lambda(A, g) - R_\lambda(A, g_e)\| + \|R_\lambda(A, g_e) - R_\lambda(\tilde{A}, g_e)\|.$$

In both cases, the operator error is often to large to address the problem just with a regularization.

Directly solving the normal equation is in general not feasible for large initial $A$. The matrix $A^*A$ is large and most often dense even if $A$ is sparse. This means directly solving the normal equation takes a lot of computation time.

### 2.2.2 Gradient descent method

Another approach is to use iterative descent algorithms, where for an $x_i$ a descent direction $d_i$ and a step length $s_i$ are chosen. The $x_i$ is then updated to $x_{i+1} = x_i + s_i \cdot d_i$. One way of choosing the direction involves the gradient of the objective function.

**Definition 2.7.** The function $F : X \to Y$

- has the *directional derivative* at point $u$ in direction $v$

$$F'(u; v) := \lim_{t \searrow 0} \frac{F(u + tv) - F(u)}{t} = f'_v(t)|_{t=0}$$

  if the limit exist;

- is *Gâteaux-differentiable* if the directional derivatives $F'(u; v)$ exist for all $v \in X$ and if

$$DF(u) : X \to Y, \quad v \mapsto F'(u; v)$$

  with $DF(u) \in \mathcal{L}(X, Y)$;

- is *Frechet-differentiable* if in addition to being Gâteaux-differentiable it also holds that

$$\lim_{\|v\| \to 0} \frac{\|F(u + v) - F(u) - DF(u)v\|_Y}{\|v\|_X} = 0.$$

*Remark* 2.8. Hilbert spaces are reflexive. This means we can also represent the derivative $F'$, which is an element of the dual space $X^*$, with an element of $X$ using the Riesz representation. We denote this element with $\nabla F$, the gradient of $F$.

If we have a minimization problem

$$\min_{x \in X} J(x), \tag{2.6}$$

where $J : X \to \mathbb{R}$ is differentiable, we can use the *gradient decent algorithm* 1

**Algorithm 1** gradient descent

1: **Input:** starting point $x_0$, step size rule
2: $x = x_0$
3: $k = 0$
4: **while** stopping criterion is not fulfilled **do**
5: choose step size $\mu_k$
6: $x = x - \mu \nabla J(x)$
7: $k = k + 1$
8: **end while**

We will prove the convergence of the proximal gradient descent method in the next section of this thesis. It can be seen as a special case of the proximal gradient descent algorithm.

## 2.3 Convex optimization

This section gives the necessary background of convex optimization, for the later theory of section 3.2. It based on [4, 2, 1].

We often want to use a non-differentiable penalty term. For example when we expect a solution with sharp edges, we would like to use L1 $\|\cdot\|_1$ or *total variation* (TV) $\|\nabla\cdot\|_1$, or we might want to add a constraint to the solution space with an indicator function,

$$\delta_C : X \to \bar{\mathbb{R}}, \quad \delta_C(x) = \begin{cases} \infty, & \text{if } x \in C \\ 0, & \text{otherwise} \end{cases}$$

with $\bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. One common use case is a positivity constraint with $C = R_+^N = \left\{x \in \mathbb{R}^N | x_i \geq 0, \forall i = 1, \ldots, N\right\}$. This means we now want to minimize

$$\min_{x \in X} J(x) = \frac{1}{2}\|Ax - y\|_X^2 + \lambda\Lambda(x),$$

but $J$ is no longer differentiable. We can define a more general type of derivative, the *subdifferential* $\partial J(x)$, which is composed of all affine linear functions below the graph of $J$ that touch $J(x)$. It is a function that maps from $X$ to the power set of $X$ $\mathcal{P}(X)$. We denote that with $X \rightrightarrows X$.

**Definition 2.9** (subdifferential)**.** Let $X$ be a linear space and $X^*$ the corresponding dual space. The subdifferential $\partial J(u) : X \rightrightarrows X$, of a function $J : X \to \bar{\mathbb{R}}$, is defined by

$$\partial J(u) = \{p^* \in X^* | J(v) \geq J(u) + (p^*|v - u)_{X^*,X} \forall v \in X\}.$$

To ensure that we are not dealing with a trivial case, we also need a $u \in X$ to exist such that $J(u) < \infty$. If this is the case , we call $J$ *proper*. We also want the subdifferential to be non-empty for all $u \in X$. To achieve this, we make two assumptions about $J$.

**Definition 2.10** (lower semi-continuity)**.** We call $J : X \to \bar{\mathbb{R}}$ lower semi-continuous if

$$J(u) \leq \liminf_{k \to \infty} J(u_k)$$

for $u_k \to u$ and $k \to \infty$.

**Definition 2.11** (convexity)**.** For $J : X \to \bar{\mathbb{R}}$,

- we call $J$ *convex*

$$J(tu + (1-t)v) \leq tJ(u) + (1-t)J(v), \quad \forall u, v \in X, t \in [0, 1].$$

- We call $J$ *strict convex* at $u$, if

$$J(tu + (1-t)v) < tJ(u) + (1-t)J(v), \quad \forall u, v \in X, t \in ]0,1[.$$

- We call $J$ *$\alpha$-strongly convex*, if

$$J(tu+(1-t)v) \leq tJ(u)+(1-t)J(v)+\frac{\alpha}{2}t(1-t)\|u-v\|^2, \quad \forall u, v \in X, t \in [0,1].$$

The strong convexity is something that helps to proof faster convergence rates. Here we will need it to show some necessary lemmas from the paper [15].

With the subdifferential we can get a necessary and sufficient optimality condition.

**Theorem 2.12** (Fermat's rule)**.** *Let $X$ be a Banach space and $J : X \to \bar{\mathbb{R}}$ be proper and convex. Then every local minimizer is a global one and $\bar{u}$ is a minimizer if and only if 0 is in the subdifferential of $J$ at $\bar{u}$, i.e.*

$$J(\bar{u}) \leq J(u) \quad \forall u \in X \quad \Leftrightarrow \quad 0 \in \partial J(\bar{u}).$$

*Proof.* Let $\bar{u}$ be a local minimizer. Then there exists an $\varepsilon > 0$ such that $J(\bar{u}) < J(\tilde{u})$ for all $\tilde{u} \in B_\varepsilon(\bar{u})$, the $\varepsilon$ ball around $\bar{u}$. Using the convexity of $J$ at $u_t = tu + (1-t)\bar{u}$ for an $u \in X$ and $t \in ]0,1[$, we get for a sufficiently small $t$ that

$$J(\bar{u}) \leq J(u_t) \leq tJ(u) + (1-t)J(\bar{u}),$$

and therefore

$$tJ(\bar{u}) = 1 - (1-t))J(\bar{u}) \leq tJ(u) \Rightarrow J(\bar{u}) \leq J(u) \quad \forall u \in X.$$

The second claim follows from

$$J(u) \geq J(\bar{u}) \quad \forall u \in X \overset{\pm 0}{\Longleftrightarrow} J(u) \geq J(\bar{u}) + (0|u-\bar{u}) \quad \forall u \in X \overset{\text{def}}{\Longleftrightarrow} 0 \in \partial J(\bar{u}).$$

□

For later proves we need two properties of the subdifferential. First, we will proof a connection between the Gâteaux derivative and the subdifferential .

**Lemma 2.13.** *If $J : X \to \bar{\mathbb{R}}$ is convex and Gâteaux-differentiable at $u$, the derivative is the only element of the subdifferential*

$$\{D\,J(u)\} = \partial J(u).$$

*Proof.* From the convexity of $J$ it follows that

$$J(u + t(v - u)) - J(u) \leq t(J(v) - J(u)).$$

If we then divide both sides by $t$ and take the limes $t \to 0$, we get the definition of the derivative

$$\mathrm{D}\, J(u)(v - u) = \lim_{t \to 0} \frac{J(u + t(v - u)) - J(u)}{t} \leq J(v) - J(u),$$

which we can rewrite as

$$J(v) \geq J(u) + \mathrm{D}\, J(u)(v - u).$$

This holds for all $v \in X$ and therefore $\mathrm{D}\, J(u) \in \partial J(u)$.

Now let $p^* \in \partial J(u)$. Then for all $t > 0$ and $v \in \mathcal{U}$ we get

$$\frac{J(u + tv) - J(u)}{t} \geq (p^*, v).$$

We take the limit $t \to 0$ on both sides and get

$$(\mathrm{D}\, J(u), v) \geq (p^*, v).$$

Subtracting the right-hand side gives us

$$(\mathrm{D}\, J(u) - p^*, v) \geq 0.$$

This still holds for all $v \in \mathcal{U}$, and that implies $p^* = \mathrm{D}\, J(u)$. □

The subdifferential also has the following sum property.

**Theorem 2.14** (Moreau Rockafellar)**.** *Let $X$ be a Banach space and $J_i : X \to \bar{\mathbb{R}}$ proper and convex for $i = 1, ..., n$. We have the sum rule*

$$\partial \left( \sum_{i=1}^{n} J_i \right)(x) = \sum_{i=1}^{n} \partial J_i(x).$$

*This holds for $n \geq 2$ if there exists a $x_0 \in X$ such that all $J_i$ are finite at $x_0$ and all but at most one of the $J_i$ are continuous at $x_0$.*

*Remark* 2.15. The $\supseteq$ inclusion follows from the definition of the subdifferential. Let $p_i^* \in \partial J_i(x)$, then we know that

$$J_i(v) \geq J_i(x) + (p_i^*, v - x), \forall v \in X, i = 1, ..., n.$$

Adding those inequalities up gives us

$$\begin{aligned} \sum_{i=1}^{n} J_i(v) &\geq \sum_{i=1} J_i(x) + \sum_{i=1}^{n} (p_i^*, v - x) \\ &= \sum_{i=1} J_i(x) + \left( \sum_{i=1}^{n} p_i^*, v - x \right), \forall v \in X \end{aligned}$$

and this implies that

$$\sum_{i=1}^{n} p_i^* \in \partial \left( \sum_{i=1}^{n} J_i \right)(x).$$

The $\subseteq$ inclusion takes more time and can be found for example in [13].

Using an element of the subdifferential as a descent direction for a descent algorithm is not feasible, because it is not always a descent direction. Instead, we will use the *proximal operator*.

**Definition 2.16** (Proximal operator)**.** For $J : X \to \bar{\mathbb{R}}$ proper, convex and lower semi-continuous and a $\mu > 0$, we define the *proximal operator* as

$$\text{prox}_{\mu J}(z) := \arg\min_{u \in X} \frac{1}{2\mu} \|u - z\|^2 + J(u)$$

*Remark* 2.17. The proximal operator is non expansive, i.e.

$$\|\text{prox}_G(u) - \text{prox}_G(v)\| \leq \|u - v\| .$$

This is proven for example in [1].

Later we will need a connection between the proximal operator and the subdifferential.

**Lemma 2.18.** *Let $J : X \to \bar{\mathbb{R}}$ be proper, convex and lower semi-continuous, $u, v \in X$ and $\mu > 0$ arbitrary, then it holds that*

$$v \in \partial J(u) \Leftrightarrow u = prox_{\mu J}(u + \mu v). \tag{2.7}$$

*Proof.* Using the optimality condition of (2.7) gives us

$$\bar{u} = \text{prox}_{\mu J}(z) \Leftrightarrow 0 \in \partial \left( \frac{1}{2\mu} \|\bar{u} - z\|^2 + J(\bar{u}) \right).$$

We can use the Moreau-Rockefellar theorem to split the subdifferential

$$\partial \left( \frac{1}{2\mu} \|\bar{u} - z\|^2 + J(\bar{u}) \right) = \frac{1}{\mu}(\bar{u} - z) + \partial J(\bar{u}). \tag{2.8}$$

This makes sense, because $X$ is a Hilbert space, this means we can identify $X^*$ with $X$ and get that $\partial J(\cdot) \subseteq X$. Rearranging (2.8), we get that

$$\bar{u} = \text{prox}_{\mu J}(z) \Leftrightarrow \frac{z - \bar{u}}{\mu} \in \partial J(\bar{u}). \tag{2.9}$$

With $\bar{u} = u$ and $z = u + \mu v$, the following applies

$$v = \frac{u + \mu v - u}{\mu} \in \partial J(u).$$

For the other direction we can do this argument in reverse. □

With that we can derive the proximal gradient method, see algorithm 2. We first assume that we can split $J$ into $F, G : X \to \bar{\mathbb{R}}$ such that

$$J(u) = F(u) + G(u),$$

this is for example possible if the assumptions of the Moreau-Rockafellar theorem 2.14 are fulfilled. We then get from the optimality condition that

$$0 \in \partial F(u) + \partial G(u) \Leftrightarrow p \in \partial F(u) \text{ and } -p \in \partial G(u).$$

From lemma 2.18 we know that $-p \in \partial G(u)$ is equivalent to

$$u = \text{prox}_{\mu G}(u - \mu p).$$

If we assume that $F$ is differentiable, then lemma 2.13 gives us that $p = \nabla F(u)$ and we can write

$$u = \text{prox}_{\mu G}(u - \mu \nabla F(u)),$$

thus we can try to find a fixed point by iterating

$$u_{k+1} = \text{prox}_{\mu G}(u_k - \mu \nabla F(u_k)).$$

This iteration scheme is called *proximal gradient descent* method.

*Remark* 2.19. If $J$ is already differentiable, we can set $F = J$ and $G \equiv 0$, then $\text{prox}_0(z) = z$, i.e., we get the gradient descent method.

**Algorithm 2** proximal gradient algorithm

1: **Input:** $x_0$, $\mu_k$-selection-rule
2: $x = x_0$
3: $k = 0$
4: **while** stopping criterion is not fulfilled **do**
5: $\quad \mu = \mu_k$-selection-rule$(k)$
6: $\quad x = \text{prox}_{\mu G}(x - \mu \nabla F(x))$
7: $\quad k = k + 1$
8: **end while**

**Example 2.20** (proximal operator of $\|\cdot\|_1$)**.** *Later we want to set our penalty term to* $\|\cdot\|_1$ *and use algorithm 2, for that we will now compute* $prox_{\lambda\|\cdot\|_1}$*. We will use* (2.9) *from the proof of lemma 2.18*

$$\bar{u} = prox_{\lambda\|\cdot\|_1}(z) \Leftrightarrow \frac{z - \bar{u}}{\lambda} \in \partial\|\cdot\|_1(\bar{u}).$$

*We can rewrite this as*

$$\bar{u} = prox_{\lambda\|\cdot\|_1}(z) \Leftrightarrow z \in \{\bar{u}\} + \lambda\partial\|\cdot\|_1(\bar{u}) \tag{2.10}$$

$$\Leftrightarrow z \in (I + \lambda\partial\|\cdot\|_1)(\bar{u}), \tag{2.11}$$

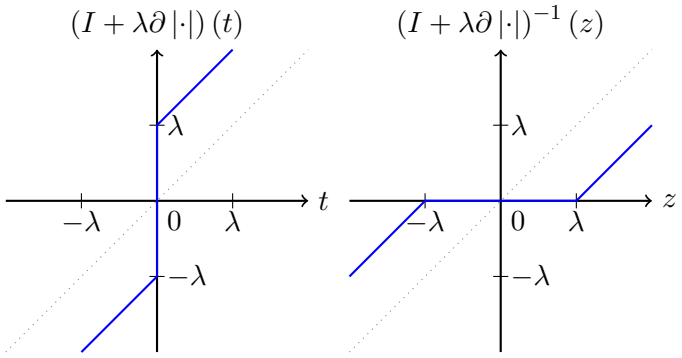



Figure 3: illustration that the soft shrinkage function is the inverse

*with the identity operator $I$. If we now can compute $(I + \lambda\partial\|\cdot\|_1)^{-1} z$, we are done. That this is invertible is demonstrated graphically in Figure 3. We can compute $\partial\|x\|_1$ with the help of the Moreau-Rockafellar theorem 2.14*

$$\partial\|x\|_1 = \partial\left(\sum_{i=1}^{N}|x_i|\right) = \sum_{i=1}^{N}\partial|x_i|.$$

*The subdifferential of $|\cdot|$ is*

$$\partial|t| = \begin{cases} 1 & \text{for } t > 0, \\ -1 & \text{for } t < 0, \\ [-1,1] & \text{for } t = 0. \end{cases}$$

*With that we can get that*

$$(I + \lambda\partial\|\cdot\|_1)^{-1}(x_i) = \underbrace{sign(t)\,[|x_i| - \lambda]_+}_{:=\mathcal{S}_\lambda(x_i)},$$

*where $[\cdot]_+$ is the projection on the positive real numbers and $S_\lambda : \mathbb{R} \to \mathbb{R}$ is called the* soft shrinkage*. Applying this to all components gives us the proximal operator for the $L^1$ norm, the so-called* soft shrinkage operator *$\$_\lambda : \mathbb{R}^N \to \mathbb{R}^N$*

$$\$_\lambda(x) = (S_\lambda(x_i))_{i=1,\dots,N} = (I + \lambda\partial\|\cdot\|_1)^{-1}(x).$$

*Remark* 2.21. For an objective function of the form

$$J(x) = \frac{1}{2}\|Ax - y\|_X^2 + \lambda\|x\|_1,$$

the algorithm 2 is the *iterative soft shrinkage algorithm* (ISTA), with update step

$$x_k + 1 = \$_\lambda\left(x_k - \lambda A^*(Ax - y)\right). \tag{2.12}$$

For the proof of convergence from algorithm 2, we need an additional lemma and that $F'$ is Lipshitz-continuous.

**Lemma 2.22.** *Let $F'$ be $L$-Lipschitz-continuous, then we get*

$$F(v) \leq F(u) + \langle F'(u), v-u\rangle + \frac{L}{2}\|v-u\|^2$$

*Proof.* From the definition of the derivative and the fact that we are in a Hilbert space, we get

$$\frac{d}{dt}F(u+t(v-u)) = \langle F'(u+t(u-v)), v-u\rangle, \quad \forall v,u \in X.$$

Integrating both sides gives us

$$\int_0^1 \langle F'(u+t(u-v)), v-u\rangle dt = F(v) - F(u).$$

Adding a zero, rearranging and using the Cauchy-Schwarz inequality gives us

$$\begin{aligned}
F(v) &= F(u) + \int_0^1 \langle F'(u+t(v-u)), v-u\rangle dt + \langle F'(u), v-u\rangle - \langle F'(u), v-u\rangle \\
&= F(u) + \int_0^1 \langle F'(u+t(v-u)) - F'(u), v-u\rangle dt + \langle F'(u), v-u\rangle \\
&\leq F(u) + \langle F'(u), v-u\rangle + \int_0^1 \underbrace{\|F'(u+t(v-u)) - F'(u)\|}_{\leq L\|t(v-u)\|} \|v-u\| dt \\
&\leq F(u) + \langle F'(u), v-u\rangle + L\int_0^1 t dt \|v-u\|^2 \\
&= F(u) + \langle F'(u), v-u\rangle + \frac{L}{2}\|v-u\|^2.
\end{aligned}$$

□

With that we can now prove the convergence of the proximal gradient descent algorithm.

**Theorem 2.23.** *Let $F$ fulfil the assumptions of lemma 2.22, let $G$ be convex and let $(x_k)_{k\in\mathbb{N}}$ be a sequence generated by algorithm 2, with $l_k < L^{-1}$ for all $k \in \mathbb{N}$. Then*

$$\lim_{k\to\infty} x_k = \bar{x},$$

*with $0 \in \partial J(\bar{x})$.*

*Proof.* From the definition of algorithm 2, we get

$$\begin{aligned} x_{k+1} &= \text{prox}_{\mu G}(x_k - \mu F'(x_k)) \\ &= \text{prox}_{\mu G}\left(x_{k+1} + \mu\left[\frac{x_k - x_{k+1}}{\mu} - F'(x_k)\right]\right). \end{aligned} \tag{2.13}$$

We define the general gradient or*prox-grad operator* $T_{\mu_k} : X \to X$ as

$$T_{\mu_k}(x) := \frac{x - \text{prox}_{\mu G}(x - \mu F'(x))}{\mu}.$$

With that we can write

$$\frac{x_k - x_{k+1}}{\mu} = T_{\mu_k}(x_k)$$

and substituting this into (2.13) gives us

$$x_{k+1} = \text{prox}_{\mu G}\left(x_{k+1} + \mu\left[T_{\mu_k}(x_k) - F'(x_k)\right]\right).$$

From lemma 2.18 we now know that $T_{\mu_k}(x_k) - F'(x_{k)} \in \partial G(x_{k+1})$. By rearranging the inequality from the subdifferential definition, we get

$$G(x_{k+1}) \leq G(z) + \langle T_{\mu_k}(x_k) - F'(x_k), x_{k+1} - z\rangle \quad \forall z \in X.$$

$F$ fulfils the assumptions of lemma 2.22, and for $v = x_{k+1}, u = x_k$ the resulting inequality is

$$F(x_{k+1}) \leq F(x_k) + \langle F'(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2.$$

Additionally, we know from lemma 2.13 that $F'(x_k) \in \partial F(x_k)$. Using the inequality from the subdifferential again, we get

$$F(x_k) \leq F(z) + \langle F'(x_k), x_k - z\rangle \quad \forall z \in X.$$

The two inequalities for $F$ combine to

$$F(x_{k+1}) \leq F(z) + \langle F'(x_k), x_k - z\rangle + \langle F'(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2.$$

Adding the inequalities for $F$ and $G$ together, and using $\mu_k \leq L^{-1}$, we get

$$\begin{aligned} J(x_{k+1}) &\leq J(z) + \langle F'(x_k), x_k - z\rangle + \langle F'(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\ &\quad + \langle T_{\mu_k}(x_k) - F'(x_k), x_{k+1} - z\rangle \\ &\leq J(z) + \langle T_{\mu_k}(x_k), x_k - z\rangle - \frac{\mu_k}{2}\|T_{\mu_k}(x_k)\|^2. \end{aligned}$$

Now we set $z = x_k$ and see that

$$J(x_k) - J(x_{k+1)} \geq \frac{\mu_k}{2}\|T_{\mu_k}(x_k)\|^2 > 0.$$

This means the sequence $(J(x_k))_{k\in\mathbb{N}}$ is non-increasing and even decreasing as long as $x_k \neq x_{k+1}$.

If we set $z = \bar{x}$ a minimizer of $J$, we get

$$\begin{aligned}
0 \leq J(x_{k+1}) - J(\bar{x}) &\leq \langle T_{\mu_k}(x_k), x_k - \bar{x}\rangle - \frac{\mu_k}{2}\|T_{\mu_k}(x_k)\|^2 \\
&= \frac{1}{2\mu_k}\left[2\langle x_k - x_{k+1}, x_k - \bar{x}\rangle - \|x_k - x_{k+1}\|^2\right] \\
&= \frac{1}{2\mu_k}\left[\|x_k - \bar{x}\|^2\right. \\
&\quad \left. - \left(\|x_k - \bar{x}\|^2 - 2\langle x_k - x_{k+1}, x_k - \bar{x}\rangle + \|x_k - x_{k+1}\|^2\right)\right] \\
&= \frac{1}{2\mu_k}\left[\|x_k - \bar{x}\|^2 - \|x_{k+1} - \bar{x}\|^2\right]. \qquad (2.14)
\end{aligned}$$

We now sum up both sides over $k = 1, ..., N$ and get a telescoping sum

$$\begin{aligned}
\sum_{k=1}^{N} J(u_k) - J(\bar{x}) &\leq \frac{1}{2\mu_{min}}\left[\|x_{k-1} - \bar{x}\|^2 - \|x_k - \bar{x}\|^2\right] \\
&= \frac{1}{2\mu_{min}}\left[\|x_0 - \bar{x}\|^2 - \|x_N - \bar{x}\|^2\right] \\
&\leq \frac{1}{2\mu_{min}}\|x_0 - \bar{x}\|^2.
\end{aligned}$$

Taking into account that $J$ is non-increasing gives us

$$J(x_N) - J(\bar{x}) \leq \frac{1}{N}\sum_{k=1}^{N} J(x_k) - J(\bar{x}) \leq \frac{1}{2N\mu_{min}}\|x_0 - \bar{x}\|^2,$$

and this converges to zero for $N \to \infty$. □

### 2.3.1 Moreau envelope

A different approach is to smooth out the non-differentiable parts of $J$. One way to achieve this is using the Moreau envelope, see [2, chapter].

**Definition 2.24** (Moreau envelope)**.** Let $J : X \to \bar{\mathbb{R}}$ be a proper, convex, lower semi-continuous function, then we call $M_J^\lambda : X \to \mathbb{R}$

$$M_J^\lambda(z) = \min_{u\in X}\left\{\frac{1}{2\lambda}\|u - z\|_X^2 + J(u)\right\}$$

the *Moreau envelope* of $J$.

Then one could show that the Moreau envelope is differentiable.

**Lemma 2.25.** *Let $J : X \to \bar{\mathbb{R}}$ be a proper, convex and lower semi-continuous function, then the derivative of its Moreau envelope $M_J^\lambda$ is given by*

$$D\, M_J^\lambda(x) = \frac{1}{\lambda}(x - prox_{\lambda J}(x)).$$

**Definition 2.26** (smoothability)**.** Let $\alpha, \beta > 0$ and let $J : X \to \bar{\mathbb{R}}$ be convex. $J$ is called $(\alpha, \beta)$-smoothable if for any $\mu > 0$ there exists a $h_\mu : X \to \bar{\mathbb{R}}$ such that

- $h_\mu(x) \leq J(x) \leq h_\mu(x) + \beta\mu$ for all $x \in X$
- the derivative of $h_\mu$ is $\frac{\alpha}{\mu}$ Lipschitz-continuous.

One can show that the Moreau envelope of a convex and $L$-Lipschitz-continuous $J$ is a $1/\mu$ -smooth approximation of $J$ with parameters $(1, L/2)$.

*Remark* 2.27. The Moreau envelope of $|\cdot|$ is given by the Huber function

$$:= \begin{cases} \frac{1}{2\gamma}t^2 & \text{if } |t|\,, \\ |t| - \frac{\gamma}{2} & \text{else }. \end{cases}$$

One can show that the Moreau envelope of a separable function is the sum of the Moreau envelopes of the elements the function separates into. This means that the Moreau envelope of the $L^1$ norm of $\mathbb{R}^n$ is given by

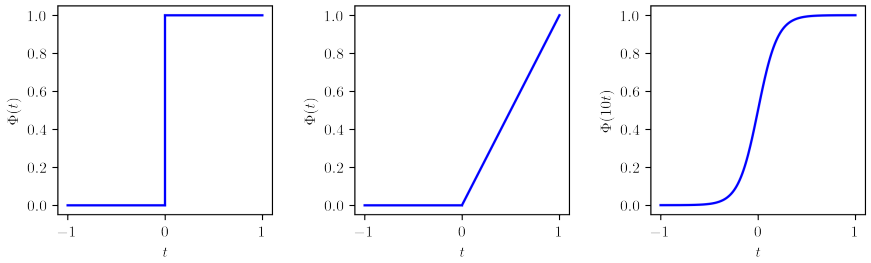
$$M^\lambda_{\|\cdot\|_1}(x) = \sum_{i=1}^{n} f_\gamma(x_i). \tag{2.15}$$

## 2.4 Machine learning

As mentioned before, we want to correct our static operator by adding and training neural networks (also referred to as artificial neural networks or machine learning models). This subsection provides a brief introduction to neural networks, based on [9, 5].

We can look at neural networks as nonlinear, parametric functions which are a combination of functions that are easy to compute. Training a neural network is then the search for parameters that result in a good approximation of our desired function.

Neural networks are structured in layers, where the output of one layer is the input of one or multiple other layers. Those outputs can also skip layers, see Figure 4. When the information also flows backwards, as indicated by the blue arrows, the model is called *recursive*. Connections that skip a layer are called *skip connections*.




(a) threshold function (b) Rectified liner unit (ReLU) (c) sigmoid

$$\Phi(t) = \begin{cases} 0 & \text{if } t < 0, \\ 1 & \text{if } t \leq 0. \end{cases} \quad \Phi(t) = \begin{cases} 0 & \text{if } t < 0, \\ t & \text{if } t \leq 0. \end{cases} \quad \Phi(t) = \frac{2}{1 + \exp(-2t)}$$

Figure 5: activation functions

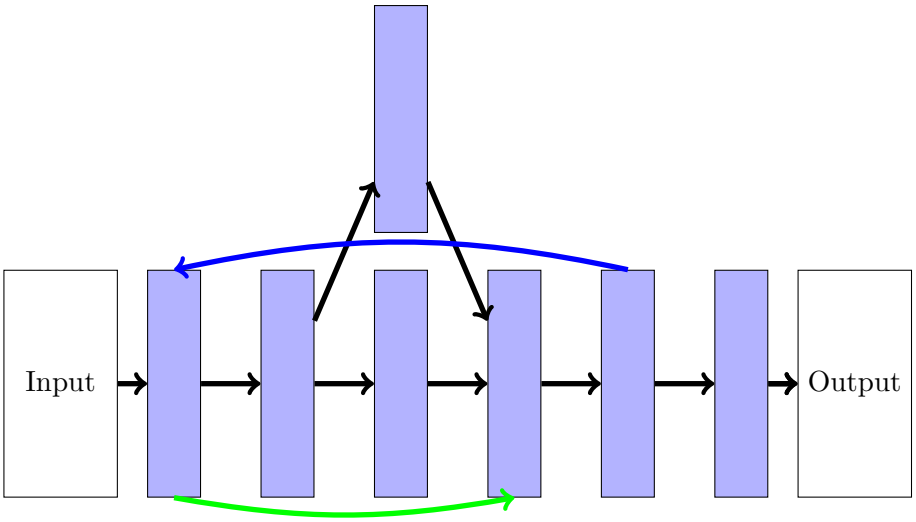



Figure 4: Neural network layers

The layers are often composed of a linear part and a nonlinear activation function. This is based on the simplified model of a biological neuron, where the inputs of a cell are summed up with weights and if the sum is larger than a threshold, a signal is transmitted. The linear part represents the combination of the inputs, and the threshold is modelled with the activation functions. In Figure 5 we can see some examples of activation functions. In reality, artificial neural networks are not at all like biological ones.

One of the most basic layers is called *dense* layer

**Definition 2.28** (dense layer)**.** Let $W \in \mathbb{R}^{N \times M}, b \in \mathbb{R}^N$ and $\Phi : \mathbb{R}^N \to \mathbb{R}^N$ a nonlinear function composed element wise of activation functions, for

$N, M \in \mathbb{N}$. We then call

$$\varphi_\vartheta : \mathbb{R}^N \to \mathbb{R}^M, \quad x \mapsto \Phi(Wx + b).$$

a *dense layer*. The elements of $W$ are called *weights* and the elements of $b$ *biases*. Together the weights and biases are the *parameters* $\vartheta$ of $\varphi_\vartheta$

The biases can be used to shift the input of the activation function, so using them it is equivalent to translating the activation function along the x-axis.

The layer is called dense because if we represent the layer as a graph where we have the input nodes on one side and the output nodes on the other side, and then connect every input node with every output node with a weighted edge. The result is a fully connected bipartite directed graph. We can not add any more edges between the input and output nodes and this is in a sense dense.

**Definition 2.29.** Convolutional layer] Let us assume we have

- a $d$-dimensional input space $X = \mathbb{R}^{n_1 \times \cdots \times n_d}$, with $n \in \mathbb{N}_0^d$ and $d \in \mathbb{N}$,
- a *kernel* $\omega \in \mathbb{R}^{m_1 \times \cdots \times m_d}$, with $m \in \mathbb{N}_0^d$,
- a *stride* $s \in \mathbb{N}^d$, and
- a $d$-dimensional output space $Y = \mathbb{R}^{n_1 - m_1 s_1 \times \ldots \times n_d - m_d s_d}$, such that $n_d - m_d s_d \in \mathbb{N}_0$.

We then have the *convolutional layer* defined as

$$\varphi_\vartheta : X \to Y, x \mapsto y = (\omega * x),$$

with

$$(\omega * x)_{i_1,\ldots,i_d} := \sum_{j=0}^{n} \omega_j x_{i \cdot s + j} = \sum_{j_1=0}^{n_1} \cdots \sum_{j_d=0}^{n_d} \omega_{j_1,\ldots,j_d} x_{i_1 s_1 + j_1,\ldots,i_d s_d + j_d}.$$

We could again add a bias $b \in Y$ and a nonlinear $\Phi : Y \to Y$ composed, element wise out of activation functions, and get

$$\varphi_\vartheta : \mathbb{R}^N \to \mathbb{R}^M, \quad x \mapsto \Phi((\omega * x) + b),$$

but this is usually not done. When we compute $\kappa \in \mathbb{N}$ different convolutions on the same input, we can stack the outputs and get an element $z = Y^\kappa$. The elements $z_i \in Y$ are then referred to as *channels*.

To have better control over the shape of $Y$, one can *crop* or add *padding* around the input $x$ before applying the constitutional layer to it.

It is called a convolutional layer because we can write $(\omega * x)$ in the form of a discretized convolution. One can also look at it as filter $\omega$ which slides over the input $x$ with stride $s$, and at each step the elements of $\omega$ are multiplied with the underlying elements of $x$ and are then added together, see Figure 6.

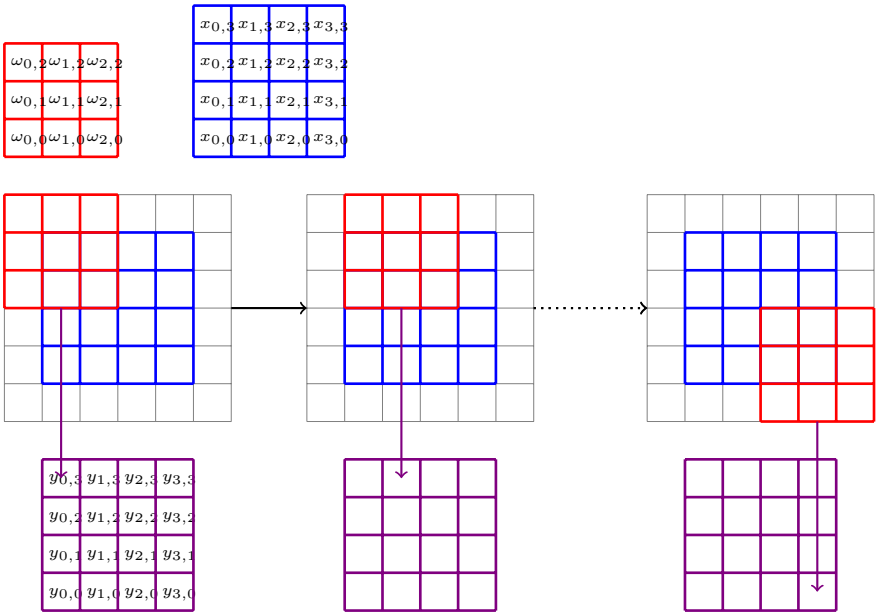


Figure 6: convolutional layer
This this is an illustration of the working of a convolutional layer, with a $4 \times 4$ input $x$, a $3 \times 3$ kernel $\omega$, a stride stride of 1 and a padding of 1, resulting in an $4 \times 4$ output $y$.

### 2.4.1 U-net

The arrangement, connections and types of layers in a neural network are called the *model architecture*. The architecture used in the paper [15] and also the main one used in this thesis is that of the U-net [21]. It is often used in neural networks that have images as in- and outputs.

The U-net, proposed in [21], is composed of convolution layers and contracting and expansive steps, see Figure 7. These resizing steps have no trainable parameters and they halve or double the side length of the images. The contracting can be done with the so-called max pool layer.

**Definition 2.30** (2D max pool)**.** The $2 \times 2$ *max-pool*

$$\text{max\_pool} : \mathbb{R}^{2N,2M} \to \mathbb{R}^{N,M}, x \mapsto y$$

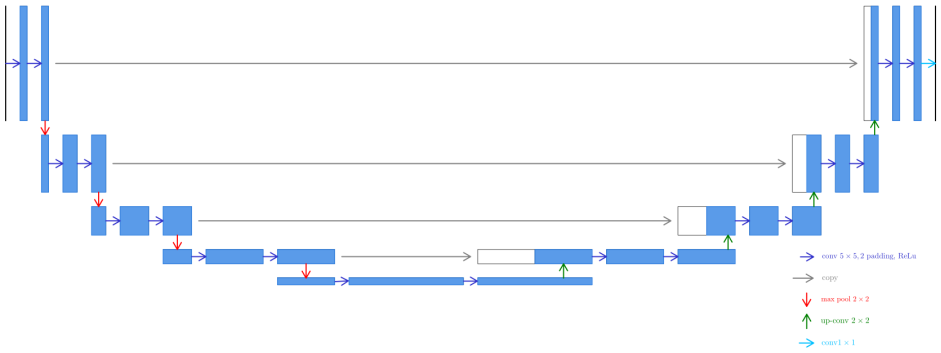
is given by

$$y_{i_1,i_2} = \max\{x_{2i_1,2i_2}, x_{2i_1+1,2i_2}, x_{2i_1,2i_2+1}, x_{2i_1+1,2i_2+1}\},$$

i.e. it divides a $2N \times 2M$-input image into a grid of $2 \times 2$-pixel blocks and creates a new $N \times M$ output image where we set the $i, j$th pixel to the highest pixel value of the $i, j$th block.

In the expansive part a $N \times M$ image is upscaled to create a $2N \times 2M$ image by interpolating the missing pixels.

Figure 7: U-net




*Remark* 2.31 (Approximation property of neural nets). From a theoretical standpoint it is shown in [5, 10, 6] that we can approximate $L^p$-functions arbitrary well if we have enough parameters and the activation functions are bounded measurable and *sigmoidal*. Sigmoidal means that they converges to 0 for $t \to -\infty$ and to 1 for $t \to \infty$.

### 2.4.2 Training

While training a neural network we want to find optimal parameters $\vartheta$ by minimizing a *loss function* $\ell : \Theta \times (X \times Y) \to \mathbb{R}_{>0}$

$$\ell\left(\vartheta(u, v)\right) = \ell\left(M_\vartheta(u), v\right),$$

with $(u, v) \in D$ where $D$ is our training dataset, which consists of the pairs of input data $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$ the output we want to achieve. For the Finite Data. One common loss function is the mean squared error (MSE)

$$\ell\left(M_\vartheta(u), v\right) = \frac{\|M_\vartheta(u) - v\|_2^2}{m},$$

for $(u, v) \in D$ with $v \in \mathbb{R}^m$. We can then define a *risk* or *cost function* $E : \Theta \to \mathbb{R}_{\leq 0}$

$$E(\vartheta) = \frac{1}{N} \sum_{i=1}^{N} \ell\left(M_\vartheta(u), v\right),$$

with $|D| = N$. In practice it is not possible to compute the minimum directly. Instead versions of the gradientn descent algorithm, for example Adam [12], are used. To compute the gradient with respect to the parameters, we need the partial derivatives for the parameters. The partial

derivatives in respect of the parameters of the earlier layers depend on the parameters of the later Layers. Using the chain rule we can work our way back from the later layers to the earlier ones. This process is called *back propagation*, see more in [9].

Computing the gradient with respect to the whole data set $D$ especially for large datasets is not feasible, instead the gradient is only computed with respect to one or few randomly chosen elements of $D$. This is called the *stochastic gradient* method. Combining the stochastic gradient with a gradient descent method then gives a *stochastic gradient descent* method. In the context of machine learning, the step size is called *learning rate*.

For more on the stochastic gradient descent see [9].

# 3 Learned operator correction

To recap the problem, we want to solve

$$Ax = y_e,$$

where $y_e = Ap + n_e$, with $A$ being the precise operator, $p$ the original phantom or object and $n_e$ some noise. If we knew the precise operator, we could try to solve our objective function

$$\mathcal{L}(x) := \frac{1}{2}\|Ax - y_e\|_Y^2 + \lambda\Lambda(x),$$

where $\lambda\Lambda$ is a penalty term, with algorithm 1 or 2 depending on conditions on $\Lambda$. The problem is that in both algorithms we need to compute

$$\nabla\frac{1}{2}\|Ax - y\|_2^2 = A^*(Ax - y_e)$$

for the update step, and we only know an imprecise operator $\tilde{A}$. Using

$$\tilde{A}^*(\tilde{A}x - y_e)$$

is not sufficient for the algorithms to get close to $p$. One approach is to correct $\tilde{A}$ and $\tilde{A}^*$. As mentioned in the introduction, the idea for this Master's thesis was inspired by the paper "On learned operator correction in inverse problems" authored by Sebastian Lunz et al. [15].

## 3.1 Operator correction in the gradient descent method

The idea put forward by Lunz et al. [15] is to train U-nets $M_\Theta$ and $M_\Phi$ as corrections for $\tilde{A}$ and $\tilde{A}^*$ and then concatenate the two to get corrected operators

$$A_\Theta = M_\Theta \circ \tilde{A} \text{ and } A_\Phi^* = M_\Phi \circ \tilde{A}^*.$$

Their hope is that if we use

$$A_\Phi^*(A_\Theta x - y_e) + \lambda \nabla \Lambda(x)$$

and

$$A^*(Ax - y_e) \approx A_\Phi^*(A_\Theta x - y_e),$$

algorithm 1 will converge to a similar solution as before. Lunz et al. then present a theory to justify this hope for the case that our objective function is differentiable. They show that if we can train our correction models well enough, we can expect that using the corrected operator, the gradient descent method reaches a neighbourhood of a minimizer of $\mathcal{L}$. We denote the objective function $\mathcal{L}$ for the precise operator, with regularization term $\lambda\Lambda(x)$

$$\mathcal{L} := \frac{1}{2}\|Ax - y\|_Y^2 + \lambda\Lambda(x),$$

and the objective function $\mathcal{L}_\theta$ for the corrected operators, with the same regularization term

$$\mathcal{L}_\Theta := \frac{1}{2}\|A_\Theta(x) - y\|_Y^2 + \lambda\Lambda(x).$$

Because we are training different corrections for the forward and the adjoint operator, we need to slightly misuse the notation

$$\nabla^\dagger \mathcal{L}_\Theta(x) := A_\Phi^*(A_\Theta(x) - y) + \lambda \nabla \Lambda(x).$$

To be able to show the main statement, we first need two lemmas.

**Lemma 3.1** (proximity to minimizer)**.** *Let $\mathcal{L}$ be strongly convex, then for all $\epsilon > 0$ there exists a $\delta > 0$ such that for all $x \in X$*

$$\mathcal{L}(x) - \mathcal{L}(\hat{x}) \leq \delta \Rightarrow \|x - \hat{x}\|_X \leq \epsilon,$$

*where $\hat{x}$ is the unique minimizer of $\mathcal{L}(x)$.*

*Proof.* From the property of strong convexity, it follows that

$$\mathcal{L}(x) \geq \mathcal{L}(\hat{x}) + \langle s_{\hat{x}}, x - \hat{x}\rangle_X + \frac{m}{2}\|x - \hat{x}\|_X^2,$$

for all $s_{\hat{x}} \in \partial\mathcal{L}(\hat{x})$. Using $0 \in \partial\mathcal{L}(\hat{x})$, we get

$$\delta \geq \mathcal{L}(x) - \mathcal{L}(\hat{x}) \geq \frac{m}{2}\|x - \hat{x}\|_X^2,$$

thus, the lemma follows with $\delta = \frac{m}{2}\epsilon^2$. □

**Lemma 3.2** (lower gradient norm bound)**.** *Let $\mathcal{L}$ be strongly convex. Then for all $\epsilon > 0$ there exists a $\delta > 0$ such that for all $x \in X$ we get that*

$$\|x - \hat{x}\|_X > \epsilon \Rightarrow \forall s \in \partial\mathcal{L}(x) : \|s\|_X > \delta,$$

*where $\hat{x}$ is the unique minimizer of $\mathcal{L}(x)$.*

*Proof.* From the property of strong convexity, it follows that

$$\mathcal{L}(x) \geq \mathcal{L}(\hat{x}) + \langle s_{\hat{x}}, x - \hat{x}\rangle_X + \frac{m}{2}\|x - \hat{x}\|_X^2,$$

for all $s_{\hat{x}} \in \partial\mathcal{L}(\hat{x})$, and using the Cauchy-Schwarz inequality we get

$$\mathcal{L}(x) - \mathcal{L}(\hat{x}) - \frac{m}{2}\|x - \hat{x}\|_X^2 \geq -\|s_{\hat{x}}\|_X\|x - \hat{x}\|_X.$$

The assumption that $\hat{x}$ is a minimizer gives us $\mathcal{L}(x) - \mathcal{L}(\hat{x}) < 0$ for $x \neq \hat{x}$. Therefore it follows that

$$\frac{m}{2}\|x - \hat{x}\|_X^2 \leq \|s_{\hat{x}}\|_X\|x - \hat{x}\|_X$$

and

$$\frac{m}{2}\|x - \hat{x}\|_X \leq \|s_{\hat{x}}\|_X,$$

which proves the lemma for $\delta = \frac{m}{2}\|x - \hat{x}\|_X$. □

To detemrine whether our correction is sufficiently trained, the parameter we look at is the alignment between the precise operator $\mathcal{L}$ and the corrected operator $\mathcal{L}_\Theta$ given by

$$\cos\Phi_\nu(x) := \frac{\langle\nabla\mathcal{L}(x), \nabla^\dagger\mathcal{L}_\Theta(x)\rangle}{\|\nabla\mathcal{L}(x)\|^2}.$$

As long as $\cos\Phi_\nu(x)$ is large enough, we can expect the gradient descent dynamics to behave in a desirable manner.

**Proposition 3.3** (convergence under alignment constraints)**.** *Let us assume that outside a neighbourhood $U$ of the minimizer $\hat{x}$ of the strongly convex functional $\mathcal{L}$*

$$\cos\Phi_\nu(x) > \delta_1 > 0,$$

*then the gradient descent dynamics over $\mathcal{L}_\Theta$ will eventually reach a neighbourhood $U$.*

*Proof.* Let $x_\Theta(t)$ be the trajectory of the gradient descent with the corrected operator such that

$$\begin{aligned}\frac{\partial}{\partial_t} x_\Theta(t) &= -\nabla^\dagger \mathcal{L}_\Theta(x_\Theta(t)) \\ &= -A_\Phi^*(A_\Theta(x_\Theta(t)) - y) + \lambda \nabla \Lambda(x_\Theta(t)).\end{aligned}$$

Then we can compute with the chain rule and the definition of the alignment that

$$\begin{aligned}\frac{\partial}{\partial_t} \mathcal{L}(x_\Theta(t)) &= \langle \nabla \mathcal{L}(x_\Theta(t)), \partial_t x_\Theta(t) \rangle_X \\ &= -\langle \nabla \mathcal{L}(x_\Theta(t)), \nabla^\dagger \mathcal{L}(x_\Theta(t)) \rangle_X \\ &= -\cos \Phi_\nu(x_\Theta(t)) \cdot \|\nabla \mathcal{L}(x_\Theta(t))\|_X^2 \\ &\leq -\delta_1 \cdot \|\nabla \mathcal{L}(x_\Theta(t))\|_X^2.\end{aligned}$$

The last inequality holds as long as $x_\Theta(t)$ is not in the neighbourhood U of $\hat{x}$. We assumed that $\hat{x} = \arg\min_{x \in X} \mathcal{L}(x)$, and because $\mathcal{L}$ is strongly convex, we can use our lemmas and get some $\delta_2$ such that

$$\|\nabla \mathcal{L}(x)\|_X \geq \delta_2 > 0 \quad \forall x \notin U.$$

Then we get

$$\partial_t \mathcal{L}(x_\Theta(t)) \leq -\delta_1 \|\nabla \mathcal{L}(x_\Theta(t))\|_X^2 \leq -\delta_1 \delta_2^2 =: c < 0.$$

This means that as long as $x_\Theta(t)$ is outside of $U$, the $\mathcal{L}(x_\Theta(t))$ decreases with a globally bounded rate, i.e. $\mathcal{L}(x_\Theta(t_0)) - c(t - t_0) \geq \mathcal{L}(x_\Theta(t))$. Together with lemma 3.1, we get that $x_\Theta(t)$ will eventually reach$U$. □

We can compute a lower bound of the alignment that helps justify the approach of the learning scheme for the corrections.

**Lemma 3.4** (complete gradient alignment bound)**.** *With our notation of $\mathcal{L}$ and $\mathcal{L}_\Theta$ we can write a lower bound for the alignment $\cos \Phi_\nu$ as*

$$\begin{aligned}\cos \Phi_\nu(x) &\geq 1 - \frac{\|\nabla^\dagger \mathcal{L}_\Theta(x) - \nabla \mathcal{L}(x)\|_X}{\|\nabla \mathcal{L}(x)\|_X} \\ &\geq 1 - \frac{\|A\|_{X \to Y} \|(A - A_\Theta)(x)\|_Y + \|(A^* - A_\Phi^*)(A_\Theta(x) - y)\|_X}{\|\nabla \mathcal{L}(x)\|_X}.\end{aligned}$$

*Proof.* We can compute that

$$\begin{aligned}\cos \Phi_\nu &= \frac{\langle \nabla \mathcal{L}(x), \nabla^\dagger \mathcal{L}_\Theta(x) \rangle_X}{\|\nabla \mathcal{L}(x)\|_X^2} \\ &= \frac{\langle \nabla \mathcal{L}(x), \nabla \mathcal{L}(x) \rangle_X}{\|\nabla \mathcal{L}(x)\|_X^2} - \frac{\langle \nabla \mathcal{L}(x), \nabla \mathcal{L}(x) - \nabla^\dagger \mathcal{L}_\Theta(x) \rangle_X}{\|\nabla \mathcal{L}(x)\|_X^2} \\ &\geq 1 - \frac{\|\nabla^\dagger \mathcal{L}_\Theta(x) - \nabla \mathcal{L}(x)\|_X}{\|\nabla \mathcal{L}(x)\|_X},\end{aligned}$$

and then the desired result follows with

$$\begin{aligned}\|\nabla^\dagger \mathcal{L}_\Theta(x) - \nabla\mathcal{L}(x)\|_X &= \|A_\Phi^*(A_\Theta(x) - y) + \lambda\nabla\Lambda(x) \\ &\quad - A^*(Ax - y) - \lambda\nabla\Lambda(x)\|_X \\ &= \|A_\Phi^*(A_\Theta(x) - y) - A^*(Ax - y)\|_X \\ &= \|A^*(Ax - y) - A^*(A_\Theta x - y) \\ &\quad + A^*(A_\Theta x - y) - A_\Phi^*(A_\Theta x - y)\|_X \\ &\leq \|A\|_{X\to Y}\|(A - A_\Theta)(x)\|_X \\ &\quad + \|(A^* - A_\Theta^*)(A_\Theta x - y)\|_X. \end{aligned} \tag{3.1}$$

□

*Remark* 3.5. The lower bound gives us a good choice for our training losses

$$\|(A - A_\Theta)(x)\|_Y$$

for the model that corrects the forward operator, and

$$\|(A^* - A_\Phi^*)(A_\Theta(x) - y)\|_X,$$

for the adjoint correction model.

Lunz et al. [15] then combine the above in the following theorem.

**Theorem 3.6** (convergence to a neighbourhood of $\hat{x}$)**.** *Let $\epsilon > 0$ and $\delta > 0$ as in lemma 3.2. Let us assume that both the adjoint and forward operator are fitted up to a $\delta/4$, i.e.*

$$\|A\|_{X\to Y}\|(A - A_\Theta)(x_n)\|_Y < \delta/4$$

$$\|(A^* - A_\Phi^*)(A_\Theta(x_n) - y)\|_X < \delta/4$$

*for all $y$ and $x_n$ during gradient descent over $\mathcal{L}$. Then eventually the gradient descent dynamics over $\mathcal{L}_\Theta$ will reach an $\epsilon$-neighbourhood $U_\epsilon$ of the accurate solution $\hat{x}$.*

*Proof.* With lemma 3.4 we can compute

$$\begin{aligned}\cos\Phi_\nu &\geq 1 - \frac{\|A\|_{X\to Y}\|(A - A_\Theta)(x)\|_Y + \|(A^* - A_\Phi^*)(A_\Theta(x) - y)\|_X}{\|\nabla\mathcal{L}(x)\|_X} \\ &\geq 1 - \frac{\delta/4 + \delta/4}{\|\nabla\mathcal{L}(x)\|_X}.\end{aligned}$$

We know that the gradient is in the subdifferential together with lemma 3.2, which gives us that $\|\nabla\mathcal{L}(x)\|_X > \delta$, if $\|x_\Theta(t) - \hat{x}\|$. And from that we get

$$\cos\Phi_\nu \geq 1 - \frac{\delta/2}{\delta} > 0.$$

With proposition 3.3 and the $\epsilon$-ball $B_\epsilon(\hat{x})$ as our neighbourhood $U_\epsilon$, we get our result. □

### 3.2 Operator correction in the proximal gradient method

We will now try to find a condition for the proximal gradient method, i.e. algorithm 2, which is similar to the one from Lunz et al.'s paper. The advantage of the proximal descent algorithm is that the penalty term does not have to be differentiable, we only need it to be proper, convex and lower semi-continuous. We write the objective function $\mathcal{L}$ for the precise operator, with the regularization term $\lambda\Lambda(x)$, as

$$\mathcal{L} := \underbrace{\frac{1}{2}\|Ax - y\|_Y^2}_{=:F(x)} + \underbrace{\lambda\Lambda(x)}_{=:G(x)},$$

the objective function $\mathcal{L}_\theta$ for the corrected operators, with the same regularization term as

$$\mathcal{L}_\Theta := \underbrace{\frac{1}{2}\|A_\Theta(x) - y\|_Y^2}_{=:F_\Theta(x)} + \underbrace{\lambda\Lambda(x)}_{=G(x)},$$

and the not quite derivative of $F_\Theta$ as

$$F_\Theta^\dagger(x) := A_\Phi^*(A_\Theta(x) - y).$$

The function we now look at instead of $\cos\Phi_\nu$ is

$$H_{\Theta,\mu}(x) := \mu\left[\frac{1}{2}\|T_{\Theta,\mu}(x)\|^2 - \langle F_\Theta^\dagger(x) - F'(x), T_{\Theta,\mu}(x)\rangle\right],$$

with $T_{\Theta,\mu} : X \to X$ given by

$$T_{\Theta,\mu}(x) = \frac{x - \operatorname{prox}_{\mu G}(x - \mu_k F_\Theta^\dagger(x))}{\mu}.$$

**Theorem 3.7.** *Let $(x_{\Theta,k})_{k\in\mathbb{N}}$ be a sequence generated by the algorithm 2 where we computed the update with the operator $F_\Theta^\dagger : X \to X$ instead of the derivative $F' : X \to X$ and with $\mu_k$ such that $\mu_k < L^{-1}$ for all $k \in \mathbb{N}$. Let us assume further that $F'$ is L-Lipschitz-continuous and let $U$ be a neighbourhood of the minimizer of $\mathcal{L}$. Then the sequence will reach $U$ if*

$$H_{\Theta,\mu_k}(x_{\Theta,k}) \geq c > 0 \tag{3.2}$$

*and all $x_{\Theta,k} \notin U$ for a positive fixed $c$.*

*Proof.* We will follow the proof of the convergence of the proximal gradient descent algorithm from theorem 2.23, but with the difference that we will

now use the sequence generated by the algorithm with the corrected operator instead of the precise one. By the definition of the algorithm, we get

$$\begin{aligned} x_{\Theta,k+1} &= \operatorname{prox}_{\mu_k G}(x_{\Theta,k} - \mu_k F_\Theta^\dagger(x_{\Theta,k})) \\ &= \operatorname{prox}_{\mu_k G}\left(x_{\Theta,k+1} + \mu_k \left[\frac{x_{\Theta,k} - x_{\Theta,k+1}}{\mu_k} - F_\Theta^\dagger(x_{\Theta,k})\right]\right). \end{aligned}$$

We define $T_{\Theta,\mu} : X \to X$,

$$T_{\Theta,\mu_k}(x) = \frac{x - \operatorname{prox}_{\mu_k G}(x - \mu_k F_\Theta^\dagger(x))}{\mu_k},$$

therefore we can write

$$\frac{x_{\Theta,k} - x_{\Theta,k+1}}{\mu_k} = T_{\Theta,\mu_k}(x_{\Theta,k})$$

and get

$$x_{\Theta,k+1} = \operatorname{prox}_{\mu_k G}\left(x_{\Theta,k+1} + \mu_k \left[T_{\Theta,\mu_k}(x_{\Theta,k}) - F_\Theta^\dagger(x_{\Theta,k})\right]\right).$$

With lemma 2.22, we get that $T_{\Theta,\mu_k}(x_{\Theta,k}) - F'(x_{k)} \in \partial G(x_{\Theta,k+1})$. By rearranging the inequality from the subdifferential definition, we get

$$G(x_{\Theta,k+1}) \leq G(z) + \langle T_{\Theta,\mu_k}(x_{\Theta,k}) - F'(x_k), x_{\Theta,k+1} - z\rangle \quad \forall z \in \mathcal{U}. \tag{3.3}$$

From lemma 2.22 we get for $v = x_{\Theta,k+1}$ and $u = x_k$ that

$$F(x_{\Theta,k+1}) \leq F(x_k) + \langle F'(x_{\Theta,k}), x_{\Theta,k+1} - x_{\Theta,k}\rangle + \frac{L}{2}\|x_{\Theta,k+1} - x_{\Theta,k}\|^2.$$

This time we do not need the $z$ term and can set $z$ in (3.3) to $x_{\Theta,k}$. Adding both inequalities up gives us, together with $\mu_k \leq L^{-1}$, that

$$\begin{aligned} \mathcal{L}(x_{\Theta,k+1}) &= F(x_{\Theta,k+1}) + G(x_{\Theta,k+1}) \\ &\leq F(x_{\Theta,k}) + \langle F'(x_{\Theta,k}), x_{\Theta,k+1} - x_{\Theta,k}\rangle + \frac{L}{2}\|x_{\Theta,k} - x_{\Theta,k+1}\|^2 \\ &\quad + G(x_{\Theta,k}) - \langle T_{\Theta\mu_k}(x_{\Theta,k}) - F_\Theta^\dagger(x_{\Theta,k}), x_{\Theta,k} - x_{\Theta,k+1}\rangle \\ &\leq \mathcal{L}(x_{\Theta,k}) + \langle F_\Theta^\dagger(x_{\Theta,k}) - F'(x_{\Theta,k}), x_{\Theta,k} - x_{\Theta,k+1}\rangle \\ &\quad + \frac{L\mu_k^2}{2}\|T_{\Theta\mu_k}(x_{\Theta,k})\|^2 - \mu_k\|T_{\Theta\mu_k}(x_{\Theta,k})\|^2 \\ &\leq \mathcal{L}(x_{\Theta,k}) + \langle F_\Theta^\dagger(x_{\Theta,k}) - F'(x_{\Theta,k}), x_{\Theta,k} - x_{\Theta,k+1}\rangle \\ &\quad - \frac{\mu_k}{2}\|T_{\Theta\mu_k}(x_{\Theta,k})\|_X^2 \\ &= \mathcal{L}(x_{\Theta,k}) - \mu_k\left[\frac{1}{2}\|T_{\Theta,\mu_k}(x_{\Theta,k})\|_X^2\right. \\ &\quad \left. -\langle F_\Theta^\dagger(x_{\Theta,k}) - F'(x_{\Theta,k}), T_{\Theta,\mu_k}(x_{\Theta,k})\rangle_X\right]. \end{aligned}$$

Together with our assumption for $H_{\Theta,\mu_k}$, we get

$$\mathcal{L}(x_{\Theta,k+1}) - \mathcal{L}(x_{\Theta,k}) \leq -c$$

as long as $x_{\Theta,k}$ is outside of $U$. This means we reduce the distance $d(U, x_{\Theta,k})$ from $x_{\Theta,k}$ to $U$ in each iteration by at least $c$. The starting distance $d(U, x_{\Theta,0})$ is finite, and therefore we reach $U$ in a finite number of iterations.

□

We would like to have an upper bound for $\|F_\Theta^\dagger(x) - F'(x)\|$ depending on the size of the neighbourhood $U$ from the previous theorem, to get something similar to theorem 3.6. For that we will first show that $\|T_\mu(x)\|$ is bounded from below by $\|x - \bar{x}\|$ and then show a lower bound for $H_{\Theta,mu}$ which is dependent on $\|T_\mu(x)\|$ instead of $\|T_{\Theta,\mu}(x)\|$.

But first, to be able to bound $\|T_\mu(x)\|$ by $\|x - \bar{x}\|$, we need to bound $\mathcal{L}(x) - \mathcal{L}(\bar{x})$ from below by $\|x - \bar{x}\|$. The next lemma gives a sufficient condition for that.

**Lemma 3.8.** *Let $\mathcal{L} : X \to \bar{\mathbb{R}}$ be a function with a convex and closed set $\mathcal{S}$ of minimizers. Let us assume that for all $N > \varepsilon > 0$, with $N > 0$ there exists an $\alpha > 0$ such that*

$$\alpha = \inf_{\{x \in X | d(\mathcal{S},x) = \varepsilon\}} \mathcal{L}(x) - \mathcal{L}(\bar{x}),$$

*with $\bar{x} \in \mathcal{S}$. From that assumption follows for convex $\mathcal{L}$ that*

$$\mathcal{L}(x) - \mathcal{L}(\bar{x}) \geq \frac{\alpha}{\epsilon} d(\mathcal{S}, x),$$

*for all $x$ such that $d(\mathcal{S}, x) \geq \varepsilon$.*

*Proof.* Let $\bar{x} \in \mathcal{S}$ be such that $\mathrm{d}(\mathcal{S}, x) = \|x - \bar{x}\|$. If we assume that the lemma does not hold, then

$$\mathcal{L}(x) < \frac{\alpha}{\varepsilon} \|x - \bar{x}\|_X + \mathcal{L}(\bar{x}).$$

We can use our condition on $\mathcal{L}$ to see that for all $x \in X$

$$\alpha \leq \mathcal{L}(\bar{x} + \frac{\varepsilon}{\|x - \bar{x}\|_X}(x - \bar{x})) - \mathcal{L}(\bar{x}).$$

We now make use of the fact that $\mathcal{L}$ is convex and get

$$\begin{aligned}
\alpha &\leq \mathcal{L}\left(\bar{x} + \frac{\varepsilon}{\|x - \bar{x}\|_X}(x - \bar{x})\right) - \mathcal{L}(\bar{x}) \\
&\leq \frac{\varepsilon}{\|x - \bar{x}\|_X}\mathcal{L}(x) + \left(1 - \frac{\varepsilon}{\|x - \bar{x}\|_X}\right)\mathcal{L}(\bar{x}) - \mathcal{L}(\bar{x}) \\
&< \alpha + \frac{\varepsilon}{\|x - \bar{x}\|_X}\mathcal{L}(\bar{x}) - \frac{\varepsilon}{\|x - \bar{x}\|_X}\mathcal{L}(\bar{x}) \\
&= \alpha
\end{aligned}$$

and $\alpha < \alpha$ is a contradiction. □

*Remark* 3.9. To fulfil the assumptions of lemma 3.8 it is enough that

- $\mathcal{L}$ is $m$ strongly convex. In that case $\mathcal{S} = \{\bar{x}\}$ and

  $$\mathcal{L}(x) - \mathcal{L}(\bar{x}) \geq \langle 0, x - \bar{x} \rangle + \frac{m}{2}\|x - \bar{x}\|_2^2,$$

  i.e. $\alpha \geq \frac{m}{2}\varepsilon^2$, this was the assumption made by Lunz et al. [15]. Alternatively, it also is enough if

- $\mathcal{L}$ is a convex and lower semi-continuous function with closed non empty set $\mathcal{S}$ of minimizers, from the convexity follows that $\mathcal{S}$ is convex.

  If we assume that the condition is not enough to fulfil the assumptions of lemma 3.8, then there exists a sequence $(x_k)_{k\in\mathbb{N}} \subset \{x \in X | \mathrm{d}(\mathcal{S}, x) = \varepsilon\}$ such that $x_k \to \tilde{x}$ for $k \to \infty$ and

  $$\liminf_{k\to\infty} \mathcal{L}(x_k) = \mathcal{L}(\bar{x}),$$

  for some $\bar{x} \in \mathcal{S}$ but from the lower semi-continuity we know that

  $$\liminf_{k\to\infty} \mathcal{L}(x_k) \geq \mathcal{L}(\tilde{x})$$

  and this, together with $(x_k)_{k\in\mathbb{N}} \subset \{x \in X | \mathrm{d}(\mathcal{S}, x) = \varepsilon\}$ is a contradiction. This means that

  $$\inf_{\{x\in X | \mathrm{d}(\mathcal{S},x)=\varepsilon\}} \mathcal{L}(x) - \mathcal{L}(\bar{x}) > 0.$$

The regularization with the $L^1$-norm fulfils the second condition: the resulting objective function $\|Ax - y\|^2 + \lambda\|c\|_1$ is convex and continuous and has closed set of minimizers.

The indicator function $\delta_C$ for a closed and convex $C$ is also convex and lower semi-continuous.

We will now use the second condition to show that $\|T_\mu(x)\|$ is bounded from below by $\|x - \bar{x}\|$.

**Lemma 3.10.** *Let $\mathcal{L}$ be convex, lower semi-continuous with a closed set of minimizers. Furthermore. let $\mathcal{L}$ and $\mu$ all fulfil the conditions of theorem 2.23 of the convergence of the proximal gradient method, then for all $\varepsilon > 0$ there exists a $\delta > 0$ such that*

$$d(\mathcal{S}, x) > \delta \Rightarrow \|T_\mu(x)\| > \varepsilon.$$

*Proof.* We know from lemma 3.8, that for some $\alpha/\varepsilon > 0$ and $\bar{x} \in \mathcal{S}$ such that $\|x - \bar{x}\| = \mathrm{d}(\mathcal{S}, x)$:

$$\mathcal{L}(x_{k+1}) - \mathcal{L}(\bar{x}) \geq \frac{\alpha}{\varepsilon}\|x_{k+1} - \bar{x}\|^2. \qquad (*)$$

From (2.14) in the proof of theorem 2.23 we get that

$$2\mu_k(\mathcal{L}(x_+) - \mathcal{L}(\bar{x})) \leq \|x - \bar{x}\|^2 - \|x_+ - \bar{x}\|^2 \qquad (**)$$

with $x_+ := x - \mu T_\mu(x)$. We make the auxiliary computation that for $a, b, c > 0$ and $c \leq a^2 - b^2$ we get that $a > b$ and

$$c \leq a^2 - b^2 = (a-b)(a+b) \leq (a+b)^2 \Rightarrow \sqrt{c} \leq a + b.$$

With that we get from $(**)$

$$\sqrt{2\mu_k(\mathcal{L}(x_+) - \mathcal{L}(\bar{x}))} \leq \|x - \bar{x}\| - \|x_+ - \bar{x}\|.$$

This means that with $(*)$ we get

$$\sqrt{\frac{\alpha}{\mu}}\|x_+ - \bar{x}\| \leq \|x - \bar{x}\| - \|x_+ - \bar{x}\|.$$

Using the reverse triangle inequality gives us

$$\begin{aligned}\sqrt{\frac{\alpha}{\mu}}\|x_+ - \bar{x}\| &\leq \|x - \bar{x}\| - \|x_+ - \bar{x}\| \\ &\leq \|x - x_+\| \\ &= \mu\|T_\mu(x)\|.\end{aligned}$$

By multiplying with $\sqrt{\mu/\alpha}$ and using the reverse triangle inequality once more, we get

$$\begin{aligned}\frac{\mu^2}{\sqrt{\alpha\mu}}\|T_\mu(x)\| &\geq \|x_+ - \bar{x}\| \\ &\geq \|x - \bar{x}\| - \mu\|T_\mu(x)\|,\end{aligned}$$

and this is equivalent to

$$\left(\frac{\mu^2}{\sqrt{\alpha\mu}} + \mu\right)\|T_\mu(x)\| \geq \|x - \bar{x}\| = \mathrm{d}(\mathcal{S}, x).$$

From that we get that for a chosen $\varepsilon > 0$, it is sufficient to use

$$\delta = \varepsilon\left(\frac{\mu^2}{\sqrt{\alpha\mu}} + \mu\right).$$

□

The second lemma shows the desired lower bound, depending now on $\|T_\mu(x)\|$ instead of $\|T_{\Theta,\mu}\|$.

**Lemma 3.11.** *The lower bound*

$$H_{\Theta,\mu}(x) \geq \frac{\mu}{2}\left[\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right]^2$$

*holds as long as*

$$\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\| \geq 0.$$

*Proof.* We compute that

$$\begin{aligned}
&\mu\|T_\mu(x) - T_{\Theta,\mu}(x)\| \\
&\quad = \left\|x - \operatorname{prox}_{\mu G}(x - \mu F'(x)) - (x - \operatorname{prox}_{\mu G}(x - \mu F_\Theta^\dagger(x)))\right\| \\
&\quad = \left\|\operatorname{prox}_{\mu G}(x - \mu F_\Theta^\dagger(x)) - \operatorname{prox}_{\mu G}(x - \mu F'(x))\right\|.
\end{aligned}$$

Making use of the fact that the proximal operator is non-expansive, we get

$$\begin{aligned}
\mu\|T_\mu(x) - T_{\Theta,\mu}(x)\| &= \left\|\operatorname{prox}_{\mu G}(x - \mu F_\Theta^\dagger(x)) - \operatorname{prox}_{\mu G}(x - \mu F'(x))\right\| \\
&\leq \left\|(x - \mu F_\Theta^\dagger(x)) - (x - \mu F'(x))\right\| \\
&= \mu\left\|F'(x) - F_\Theta^\dagger(x)\right\|.
\end{aligned}$$

Apply this to estimate

$$\|T_{\Theta,\mu}(x)\| \geq \|T_\mu(x)\| - \|T_\mu(x) - T_{\Theta,\mu}(x)\| \geq \|T_\mu(x)\| - \left\|F_\Theta^\dagger(x) - F'(x)\right\|.$$

Using this gives us

$$\begin{aligned}
\frac{2}{\mu}H_{\Theta,\mu}(x) &= \|T_{\Theta,\mu}(x)\|\left[\|T_{\Theta,\mu}(x)\| - 2\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right] \\
&\geq \|T_{\Theta,\mu}(x)\|\left[\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right].
\end{aligned}$$

If we now assume that $\|T_\mu(x)\| \geq 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|$, we get that

$$\begin{aligned}
\frac{2}{\mu}H_{\Theta,\mu}(x) &\geq \|T_{\Theta,\mu}(x)\|\left[\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right] \\
&\geq \left[\|T_\mu(x)\| - \left\|F_\Theta^\dagger(x) - F'(x)\right\|\right]\left[\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right] \\
&\geq \left[\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right]\left[\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right] \\
&\geq \left[\|T_\mu(x)\| - 3\left\|F_\Theta^\dagger(x) - F'(x)\right\|\right]^2.
\end{aligned}$$

□

We are now combining the results for the convergence of the proximal gradient method, with an corrected operator, and get something similar to theorem 3.6.

**Theorem 3.12.** *Let the assumptions of lemma 3.10 and theorem 3.7 be fulfilled . Then for all $\varepsilon > 0$ exist a $\delta$ such that the algorithm 2 will reach $\{x|d(\mathcal{S}, x) = \varepsilon\}$ around the minimizers of $\mathcal{L}$, if*

$$\frac{\delta - c}{3} \geq \left\| F_\Theta^\dagger(x) - F'(x) \right\|$$

*for all $x \notin \{x|d(\mathcal{S}, x) \leq \varepsilon\}$ and an arbitrary but fixed $c > 0$.*

*Proof.* Let us assume that for some $\tilde{c} > 0$

$$\frac{1}{2}\mu \left[ \|T_\mu(x)\| - 3 \left\| F_\Theta^\dagger(x) - F'(x) \right\| \right]^2 \geq \tilde{c} > 0,$$

this is equivalent to

$$\frac{\|T_\mu(x)\| - c}{3} \geq \left\| F_\Theta^\dagger(x) - F'(x) \right\|, \tag{3.4}$$

with $\sqrt{\frac{2\tilde{c}}{\mu}}$. This is stronger than the assumption on lemma 3.11, because

$$\|T_\mu(x)\| - 3 \left\| F_\Theta^\dagger(x) - F'(x) \right\| \geq c > 0. \tag{3.5}$$

From the lemma then follows that $H_{\Theta,\mu} \geq \tilde{c}$. This means that if (3.5) holds we are fulfilling the assumptions of theorem 3.7, i.e if we can show that (3.5) holds for all $x \notin \{x|\mathrm{d}(\mathcal{S}, x) \leq \varepsilon\}$, algorithm 2 will reach $\{x|\mathrm{d}(\mathcal{S}, x) \leq \varepsilon\}$. We can use lemma 3.10 to find a $\delta$ for our $\varepsilon$ such that

$$x \notin \{x|\mathrm{d}(\mathcal{S}, x) = \varepsilon\} \Rightarrow \|T_\mu(x)\| \geq \delta.$$

With that we get that algorithm 2 will reach $\{x|\mathrm{d}(\mathcal{S}, x) \leq \varepsilon\}$ if

$$\frac{\delta - c}{3} \geq \left\| F_\Theta^\dagger(x) - F'(x) \right\|$$

holds for all $x \notin \{x|\mathrm{d}(\mathcal{S}, x) \leq \varepsilon\}$. □

*Remark* 3.13. If $\mathcal{L}$ has a unique minimizer $\bar{x}$ then the neighbourhood from theorem 3.12 is the $\varepsilon$ Ball around $\bar{x}$

$$\{x|\mathrm{d}(\mathcal{S}, x) \leq \varepsilon\} = \mathrm{B}_\varepsilon(\bar{x}).$$

*Remark* 3.14. Theorem 3.12 it is not necessary that $F_\Theta^\dagger$ is coming from corrections for an imprecise forward operator and its adjoint. This means we can replace $F_\Theta^\dagger$ with an arbitrary function $f : X \to \mathbb{R}$ and $T_{\Theta,\mu}$ with

$$T_{\Theta,\mu}(x) = \frac{x - \operatorname{prox}_{\mu G}(x - \mu_k F_\Theta^\dagger(x))}{\mu}$$

and as long as the conditions of theorem 3.12 are fulfiled we still reach a neighbourhood $U$ of the minimizers of $\mathcal{L}$.

We can bound $\left\| F_\Theta^\dagger(x) - F'(x) \right\|$ by terms we can try to minimize in the training of the corrections.

**Lemma 3.15.** *We have that*

$$\left\| F'(x) - F_\Theta^\dagger(x) \right\|_X \leq \|A\|_{X\to Y} \|(A - A_\Theta)(x)\|_X + \|(A^* - A_\Theta^*)(A_\Theta x - y)\|_X.$$

*Proof.* By using the Cauchy-Schwarz inequality and the definition of the operator norm, we get

$$\begin{aligned}
\left\| F'(x) - F_\Theta^\dagger(x) \right\|_X &= \|A^*(Ax - y) - A_\Phi^*(A_\Theta(x) - y)\|_X \\
&= \|A^*(Ax - y) - A^*(A_\Theta(x) - y) \\
&\quad + A^*(A_\Theta x - y) - A_\Phi^*(A_\Theta(x) - y)\|_X \\
&\leq \|A\|_{X\to Y} \|(A - A_\Theta)(x)\|_X \\
&\quad + \|(A^* - A_\Theta^*)(A_\Theta x - y)\|_X.
\end{aligned}$$

□

## 3.3 Comparison

We will now try to compare the two theories. As mentioned in remark 2.19, the gradient descent algorithm is a special case of the proximal gradient method. Let us set $F = \mathcal{L}$ and $G = 0$. Then we can compute

$$\begin{aligned}
T_{\Theta,\mu_k}(x) &= \frac{x - \operatorname{prox}_{\mu_k G}(x - \mu_k F_\Theta^\dagger(x))}{\mu_k} \\
&= \frac{x - (x - \mu_k F_\Theta^\dagger(x))}{\mu_k} \\
&= \frac{\mu_k F_\Theta^\dagger(x)}{\mu_k} \\
&= F_\Theta^\dagger(x).
\end{aligned}$$

And substituting this in 3.5 gives us

$$\frac{\|\nabla \mathcal{L}\| - \sqrt{\frac{2c}{\mu}}}{3} \geq \left\| \nabla^\dagger \mathcal{L}_\Theta(x) - \nabla \mathcal{L}(x) \right\|. \tag{3.6}$$

From the lower bound 3.4 for $cos\Phi_\theta$ we get that we need

$$1 - \frac{\|\nabla^\dagger \mathcal{L}_\Theta(x) - \nabla\mathcal{L}(x)\|_X}{\|\nabla\mathcal{L}(x)\|_X} \geq c,$$

this can be reformulated to

$$(1-c)\,\|\nabla\mathcal{L}\| \geq \left\|\nabla^\dagger \mathcal{L}_\Theta(x) - \nabla\mathcal{L}(x)\right\|. \tag{3.7}$$

In comparing (3.6) to (3.7), we can ignore the terms that are dependent on $c$, because we can choose $c$ small enough to make them negligible. This gives us that for the proximal gradient method theory we need the difference of the gradients is one-third smaller than for the theory for the gradient descent method.

# 4 Numerical results

## 4.1 Implementation of the training

We will now consider how the training set is created. As seen in lemmas 3.4 and 3.15, for a convergence to $\mathrm{B}_\varepsilon(\bar{x})$ with $\varepsilon$ as small as possible, we want to minimize

$$\|(A - A_\Theta)(x)\|_Y \text{ and } \|(A^* - A_\Phi^*)(A_\Theta(x) - y)\|_X. \tag{4.1}$$

This is useful for creating training datasets if we have access to the precise operator $A$ as was the case in [15]. For the training scheme in this Master's thesis, I only used an analytic data set, i.e. I also had access to the precise operator. The ray transform is a linear operator, this means we can add it to the network without creating any additional problems in the backtracking. It can be viewed as a non-trainable dense layer, where $W$ is the Radon matrix, the bias is zero and the activation function is the identity. Thus we can build the training set for the forward model with elements of the form

$$[\text{input: } x,\ \text{output: } Ax].$$

We then use the trained forward model to compute $r = A_\Theta(x) - y$ and construct the elements of the training set for the adjoint operator correction as

$$[\text{input: } r,\ \text{output: } A^*r].$$

The question remains which $x$ to choose. In [15], the authors found that it is not enough to train the models only on initial values $x_0 = A^*_{static}y_e$. We also need to train on later iteration $x_i$. The problem is if the training set contains all iterations $i \leq N_{iter}$ computed with the corrected operators, the training is unstable, possibly because the later iterations are far away from

the iterations we would get from the precise operators. Lunz et al. [15] name two solutions, one is to use $x_i$ computed with the precise operators, but this might lead to errors accumulating over the iterations. The other solution is to build the training set over time. For that we have to recurrently compute $x_i$ and train our model. The training scheme can be seen in algorithm 3. This leads to an increase of the training sets size $s_k$ by the initial size $s_0$ in each iteration, i.e.

$$s_k = \sum_{i=0}^{k} k \cdot s_0 = \frac{k \cdot (k+1)}{2}.$$

This means the training time depends quadratically on the number of iterations. In an attempt to keep the training time in linear dependance of the number of iterations, I always trained with the newest $x_i$ and picked at random from the rest of the $x_i$ until hitting an upper bound.

To prevent overfitting, I also renewed the initial $x_0$ before computing the $x_i$ for the training set with the next iteration included.

We want to be able to use our correction on shifts we have not trained on. To do this, in every iteration we also use new operators or select randomly out of a larger pool.

This then leads to the training scheme described in algorithm 4.

*Remark* 4.1. We use $x_0 = \tilde{A}^* y_e$ as starting points and not $x_0 = 0_X$, because if we use $x_0 = 0_X$, the forward model would be trained on a training set entirely composed of $[0_X, 0_Y]$, which would lead the training in the direction of the constant zero function. On the other hand, not having a zero element in the training data leads to $A_\Theta(0_X) \neq 0_Y$ and $A^*\Phi(0_Y) \neq (0_X)$, so it might be a good idea to add some zero elements into the training data.

To compute the precise operators that include the shifts, I used the Operator Discretization Library (ODL), https://github.com/odlgroup/odl. This Python library offers a tool where one can enter a beam geometry and add for each measurement angle a shift in the detector to source point axis $u$ and one in the axis $v$ that is orthogonal to $u$, see Figure 9. The shifts I used are of the form

$$u_{shift}(angle) = \frac{1}{n} \sum_{i=0}^{n} a_{u,i} \cdot \sin(b_{u,i} \cdot angle + c_{u,i}),$$

and

$$v_{shift}(angle) = \frac{1}{n} \sum_{i=0}^{n} a_{v,i} \cdot \sin(b_{v,i} \cdot angle + c_{v,i}),$$

with parameters $a_{u,i}, b_{u,i}, c_{u,i}$ for amplitude, frequency and frequency shift, respectively.

For the implementation and training of the models I used the open-source machine learning library PyTorch, https://pytorch.org/. In my efforts to

**Algorithm 3** training scheme

**choose** phantoms $\{p^k | k \in I_k \subset \mathbb{N}\}$, precise operators $\{A^l | l \in I_l \subset \mathbb{N}\}$, starting points $\{x_0^{k,l} | (k,l) \in I_{k,l} \subseteq I_k \times I_l\}$, regularization parameter $\lambda$ and penalty term $\Lambda$
**compute** noisy data $y_e^{k,l} = A^l p^k + n_e$, with noise $n_e$ for $(k,l) \in I_{k,l}$
**for** $i = 1, \ldots, N_{iter}$ **do**
    $\nabla^\dagger F_\Theta(x) := A_\Phi^*(A_\Theta(x) - y_e^{k,l}))$
    **for** $(k,l) \in I_{k,l}$ **do**
        **for** $j = 1, \cdots, i$ **do**
            $d = \text{DescentDirection}(x_{j-1}^{k,l}, \nabla^\dagger F_\Theta(x_{j-1}^{k,l}), j, \text{algorithmType})$
            $x_j^{k,l} = x_{j-1}^{k,l} + d$
        **end for**
    **end for**
    **make** forward training set
        $\mathcal{T}_{fw} \left\{ [x_i^{k,l}, A^l x_i^{k,l}] \middle| (k,l) \in I_{k,l}, i = 0, \cdots, i \right\}$
    **for** $0 \leq t <$number of forward epochs **do**
        **train** $A_\Theta^*$ on $\mathcal{T}_{fw}$
    **end for**
    **make** adjoint training set
        $\mathcal{T}_{adj} \left\{ [r, A^{l*} r] \middle| r = A_\Theta(x) - y_e^{k,l}, (k,l) \in I_{k,l}, i = 0, \cdots, i \right\}$
    **for** $0 \leq t <$number of adjoint epochs **do**
        **train** $A_\Phi^*$ on $\mathcal{T}_{adj}$
    **end for**
**end for**

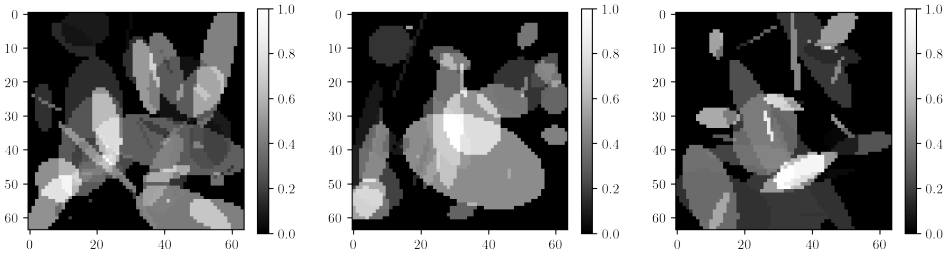
**function** DescentDirection($x, \nabla F(x), i$,algorithmType)
    **if** algorithmType == GradientDescent **then**
        $\mu = \mu_i$-selectionRule(i)
        **return** $-(\mu \nabla F(x) + \lambda \nabla \Lambda(x))$
    **else if** algorithmType == ProximalGradientDescent **then**
        $\mu = \mu_i$-selectionRule(i)
        **return** $\text{prox}_{\mu\lambda\Lambda}(x - \mu_i \nabla F(x))$
    **end if**
**end function**

**Algorithm 4** reduced and randomized training scheme

**choose** phantoms $\{p^k | k \in I_k \subset \mathbb{N}\}$, precise operators $\{A^l | l \in I_l \subset \mathbb{N}\}$, starting points $\{x_0^{k,l} | (k,l) \in I_{k,l} \subseteq I_k \times I_l\}$, regularization parameter $\lambda$ and penalty term $\Lambda$, maxLength, selectionRule
**for** $i = 1, \ldots, N_{iter}$ **do**
  **renew** $\{A^l | l \in I_l \subset \mathbb{N}\}$
  **renew** $\{p^k | k \in I_k \subset \mathbb{N}\}$
  **compute** noisy data $y_e^{k,l} = A^l p^k + n_e$, with noise $n_e$ for $(k,l) \in I_{k,l}$
  $\nabla^\dagger F_\Theta(x) := A_\Phi^*(A_\Theta(x) - y_e^{k,l}))$
  **for** $(k,l) \in I_{k,l}$ **do**
    **for** $j = 1, \cdots, i$ **do**
      $d = \text{DescentDirection}(x_{j-1}^{k,l}, \nabla^\dagger F_\Theta(x_{j-1}^{k,l}), j, \text{algorithmType})$
      $x_j^{k,l} = x_{j-1}^{k,l} + d$
    **end for**
  **end for**
  **make** forward training set
    $\mathcal{T}_{fw} \left\{ [x_i^{k,l}, A^l x_i^{k,l}] \middle| (k,l) \in I_{k,l}, i = 0, \cdots, i \right\}$
  **for** $0 \leq t <$ number of forward epochs **do**
    **train** $A_\Theta^*$ on SubSetSelection($\mathcal{T}_{fw}$,maxLength,selectionRule)
  **end for**
  **make** adjoint training set
    $\mathcal{T}_{adj} \left\{ [r, A^{l^*} r] \middle| r = A_\Theta(x) - y_e^{k,l}, (k,l) \in I_{k,l}, i = 0, \cdots, i \right\}$
  **for** $0 \leq t <$ number of adjoint epochs **do**
    **train** $A_\Phi^*$ on SubSetSelection($\mathcal{T}_{adj}$,maxLength,selectionRule)
  **end for**
**end for**

**function** SubSetSelection($\mathcal{T}$,maxLength,selectionRule)
  **if** selectionRule == fromAll **then**
    **choose** $\mathcal{S} \subseteq \mathcal{T}$ at random s.t. $|\mathcal{S}| = \min\{maxLength, |\mathcal{T}|\}$
  **else if** selectionRule = LastAndRandom **then**
    $\mathcal{T}_{last} \subseteq \mathcal{T}$ all training points that are computed with the $x_i^{k,l}$ of the latest iteration
    $\mathcal{T}_{rest} = \mathcal{T} - \mathcal{T}_{last}$
    **choose** $\mathcal{S}_{rest} \subseteq \mathcal{T}_{rest}$ at random s.t. $|\mathcal{S}_{rest}| = \min\{\max\{maxLength - |\mathcal{T}_{last}|, 0\}, |\mathcal{T}_{rest}|\}$
    $\mathcal{S} = \mathcal{T}_{last} + \mathcal{S}_{rest}$
  **end if**
  **return** $\mathcal{S}$
**end function**

Figure 8: phantoms



obtain the adjoint of the ray transform, I found that it was unfortunately not possible to use the *cuda* compatible built-in adjoint of the ODL ray transform, because it is a discretization of the back projection. While this is the adjoint of the ray transform in the $L^2$-function-space, in the discretised form it is not the adjoint in $\mathbb{R}^N$.

My workaround was to use the CPU implementation of the ray transform to extract the Radon matrix, which then enabled me to use the transposed of the matrix as the adjoint operator. However, this led to some further problems. PyTorch has no built-in method to use sparse matrices with *cuda*, and due to time constraints I had to resort to using dense matrices. The combination of the extraction process and the use of dense matrices limited me to small matrices, i.e small phantoms.

Like Lunz et al. [15], I chose U-nets for the corrections, see Figure 7. To enforce that the input size of the U-net would be the same as the output size, I needed the two dimensions of the input to have $2^4$ as a factor, so that the dimensions would be divisible by two in all four descending steps. To ensure that the input size and the output size of the individual convolutional layers with a $5 \times 5$ kernel would be the same, I used a padding of two pixels around the input. I also experimented with the positioning of the U-net with respect to $\tilde{A}$ and $\tilde{A}^*$, i.e. the correction before, after or on both sides of the operators.

## 4.2 Experimental setup

Now for the setup of the numerical experiments. I used $64 \times 64$ pixel phantoms containing 50 random ellipses, see Figure 8. To get a good reconstruction with the precise operator I choose a beam geometry setup with 256 angles and 96 detector points the proportions you can see in figure 9, additionaly you can see the size of the phantom green square and the maximal area the phantom can wobble in orange.

The shifts are as mentioned structured like

$$u_{shift}(angle) = \frac{1}{n} \sum_{i=0}^{n} a_{u,i} \cdot \sin(b_{u,i} \cdot angle + c_{u,i}),$$

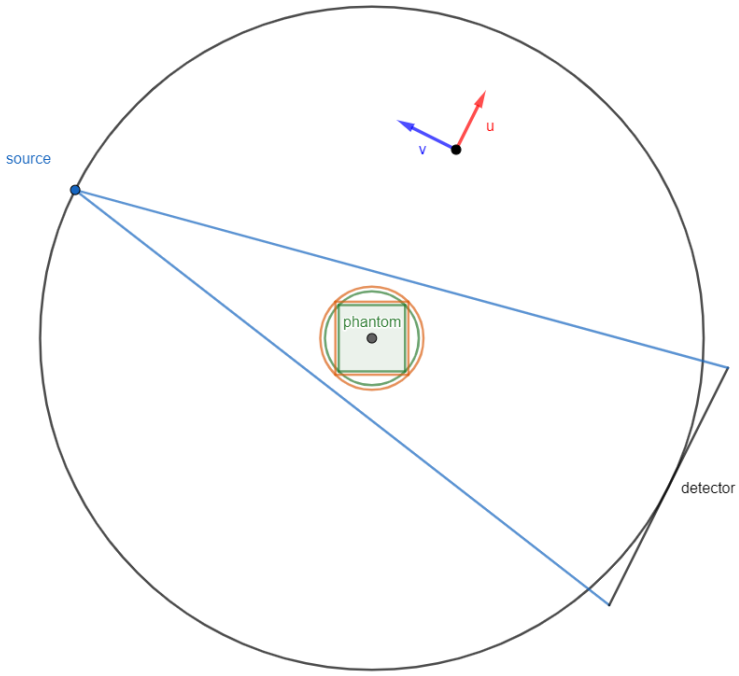



Figure 9: beam geometry in experiments

$$v_{shift}(angle) = \frac{1}{n}\sum_{i=0}^{n} a_{v,i} \cdot \sin(b_{v,i} \cdot angle + c_{v,i}),$$

there the parameters are chosen uniform at random out of intervals $a_{u,i}, a_{v,i} \in [0.03, 0.05], b_{u,i}, b_{v,i} \in [500, 5000], c_{u,i}, c_{v,i} \in [0, 2\pi]$ and $n$ is depending on the experiment between one and five. We then use the shifts to generate the precise operators to train on. In Figure 10 we can see how the sinogramm looks for no shift, a shift with $n = 1$, one with a larger amplitude and one with $n = 5$. We can see the sinus waves clearly in the sinogramms with $n = 1$ and even in the case of $n = 5$ we can imagine that we can extract information about the shifts from the sinogram. For the regularization we look at the objective functions

$$\underbrace{\frac{1}{2\lambda}\left\|A_\Theta x - y_e\right\|}_{F} + \underbrace{\|x\|_1}_{G}.$$

For the reconstruction I used ISTA with $\text{prox}_\mu G(x) = \$_{\mu\lambda}(x)$, the soft shrinkage operator as in example 2.20. For simplicity we chose a constant step size $\mu$ for all iterations and is we set $\lambda = 0.001$, because it gives good results for reconstructing with the precise operator.

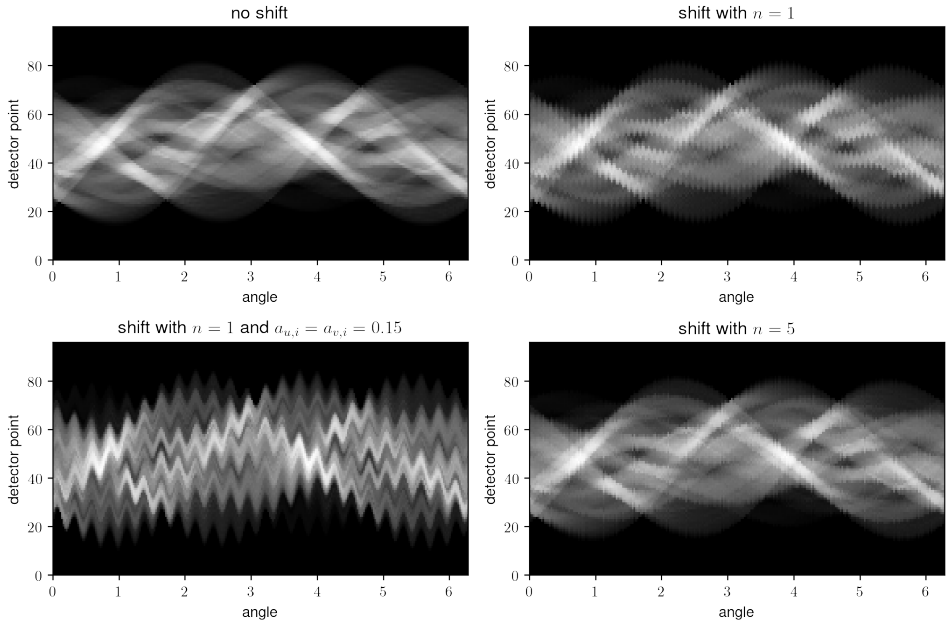



Figure 10: sinograms for different shifts with parameters $a_{u,i}, a_{u,i} \in [0.03, 0.05], b_{u,i}, b_{u,i} \in [500, 5000], c_{u,i}, c_{u,i} \in [0, 2\pi]$

### 4.3 Numerical experiments

I experimented with the positioning of the corrections.

Lunz et al. [15] place the U-net corrections $M_\Theta$ and $M_\Phi$ after the imprecise operators $\tilde{A}$ and $\tilde{A}^*$

$$A_\Theta := M_\Theta \circ \tilde{A}, \quad A^*_\Phi := M_\Phi \circ \tilde{A}^*,$$

where $\circ$ is the concatenation operator, i.e. $M_\Theta \circ \tilde{A}(x) = M_\Theta(\tilde{A}(x))$. This results in the descent direction, in algorithm 2, being computed with

$$M_\Phi \left( \tilde{A}^* \left( M_\Theta \left( \tilde{A}x \right) - y \right) \right)$$

- positioning 1 is as it was done in [15], i.e the corrections come after the operators, as in (4.3),
- positioning 2 is exactly opposite to that, i.e. the corrections are placed before the operator

$$A_\Theta := \tilde{A} \circ M_\Theta, \quad A^*_\Phi := \tilde{A}^* \circ M_\Phi,$$

- positioning 3 is the positioning in the data space only

$$A_\Theta := M_\Theta \circ \tilde{A}, \quad A^*_\Phi := \tilde{A}^* \circ M_\Phi,$$

- positioning 4 is the positioning in the phantom space only

$$A_\Theta := M_\Theta \circ \tilde{A}, \quad A_\Phi^* := M_\Phi \circ \tilde{A}^*$$

and

- positioning 5 sandwiches the operators between U-nets

$$A_\Theta := M_{\Theta_2} \circ \tilde{A} \circ M_{\Theta_1}, \quad A_\Phi^* := M_{\Phi_2} \circ \tilde{A}^* \circ M_{\Phi_1}.$$

All U-nets use four contracting and expanding steps. The convolutional layers used have $5 \times 5$ kernels, a stride of 1 and a padding of 2, see Figure 7.

I trained the corrections with positionings 1,2,3 and 4 on the ISTA algorithm on 50 iterations with 5 operators which used a shift with parameters $a_{u,i}, a_{u,i} \in [0.03, 0.05], b_{u,i}, b_{u,i} \in [500, 5000], c_{u,i}, c_{u,i} \in [0, 2\pi]$ and $n = 1$. Of the operators four are chosen from a list of 100 operators, and created new. For each operator 100 phantoms where used with a maximum of training set size of 5000. This means after the 10th iteration the training set is reduced (see the training scheme in algorithm 4). The correction with positioning 5 was trained on operators with the same shifts on 100 iterations and a maximal training set size of 2000.

Due to an error on my side, the constant step size $\mu = 0.0004$ in training was chosen a bit too large to fulfil the criterion of being smaller than the inverse of the Lipschitz constant $L$ of $F'$. The corrections are trained on $L^{-1} \approx 0.003$. This means that the theory does not guaranties us convergence. In Figure 11, we see the progress $\|\mathcal{L}(x_i) - \mathcal{L}_{i+1}\|$ the iteration made towards the minimum of $\mathcal{L}$ and $H_{\Theta,\mu}(x_i)$, both are scaled with $\|T_{\Theta,\mu}(x_i)\|^2$ so we can see better if $\|\mathcal{L}(x_i) - \mathcal{L}_{i+1}\| \leq H_{\Theta,\mu}(x_i)$. This inequality should if the assumptions of theorem 3.7 are fulfiled. That $L^{-1} \geq \mu$ is one of the assumption and we can see that not fulfiling it leads our case to the fact that the inequality does not hold for early iterates.

This led me to discover that the model is tolerant for changes in $\mu$, see Figure 14. The rest of the given examples are also computed with $\mu = 0.002$.

In Figure 12 we see the same functions as in 11 with the only difference that $\mu = 0.002$, we see now that the assumptions of 3.7 are fulfiled, we get that the inequality $\|\mathcal{L}(x_i) - \mathcal{L}_{i+1}\| \leq H_{\Theta,\mu}(x_i)$ holds.

Figure 14 shows the results obtained from using the corrected operator for a shift from the list the corrections where trained on. We see the reconstructions after 49 iterations using ISTA with $\mu = 0.0002$.

Figure 14 also shows the loss over the first 600 iterations. We can see that at the beginning all corrections are better than the static operator, but that losses reach their minima before 100 iterations and then start to increase again. The one that has the best loss is the sandwich correction. This is not surprising, since it has twice as many trainable parameters as the other correction and was trained for the most iterations. The next best

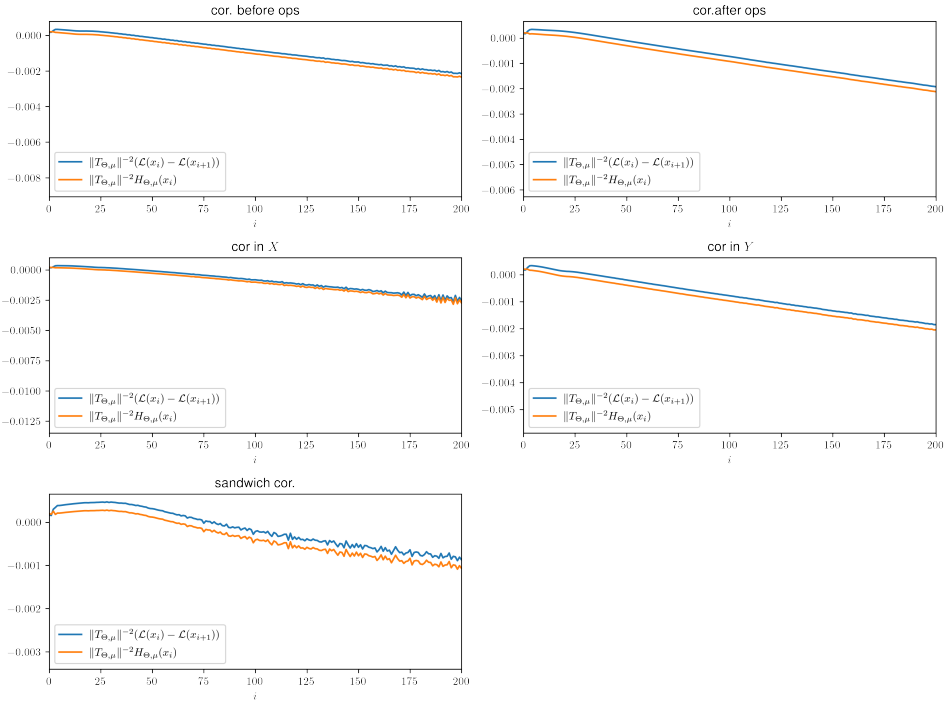



Figure 11: $\|\mathcal{L}(x_i) - \mathcal{L}_{i+1}\| \leq H_{\Theta,\mu}(x_i)$ for to large $\mu = 0.0004$

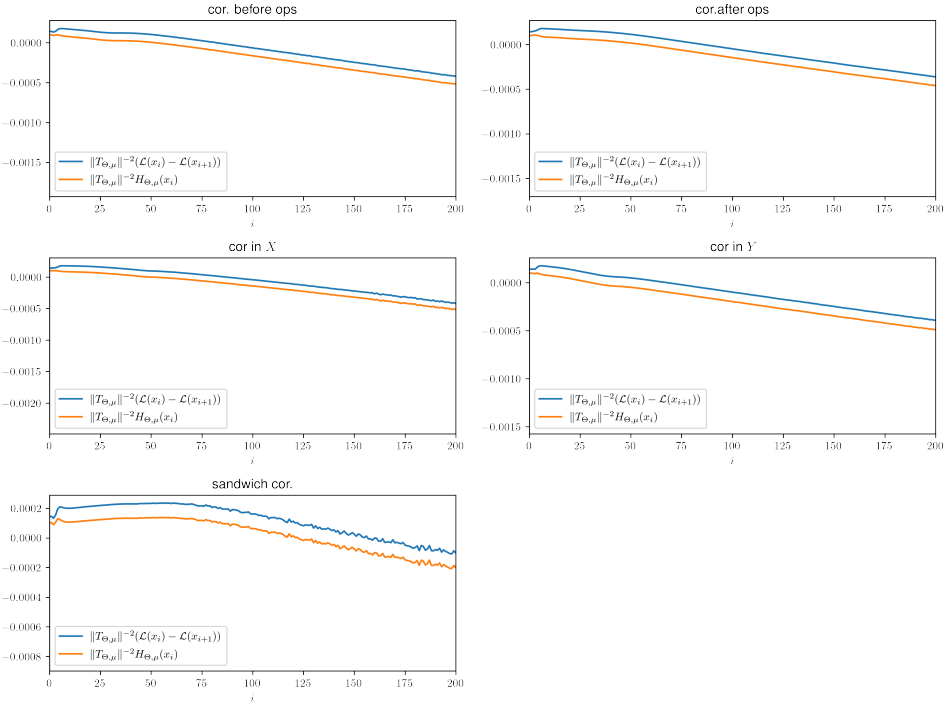



Figure 12: $\|\mathcal{L}(x_i) - \mathcal{L}_{i+1}\| \leq H_{\Theta,\mu}(x_i)$ for theory conform $\mu = 0.0002$

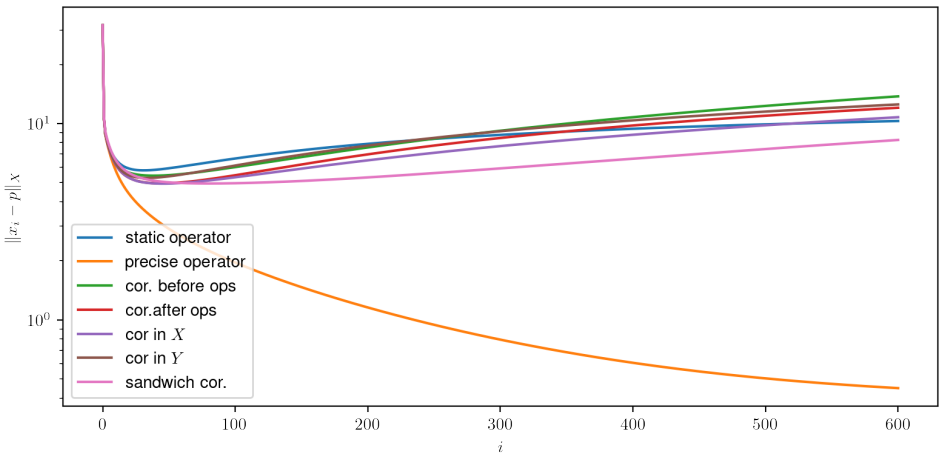


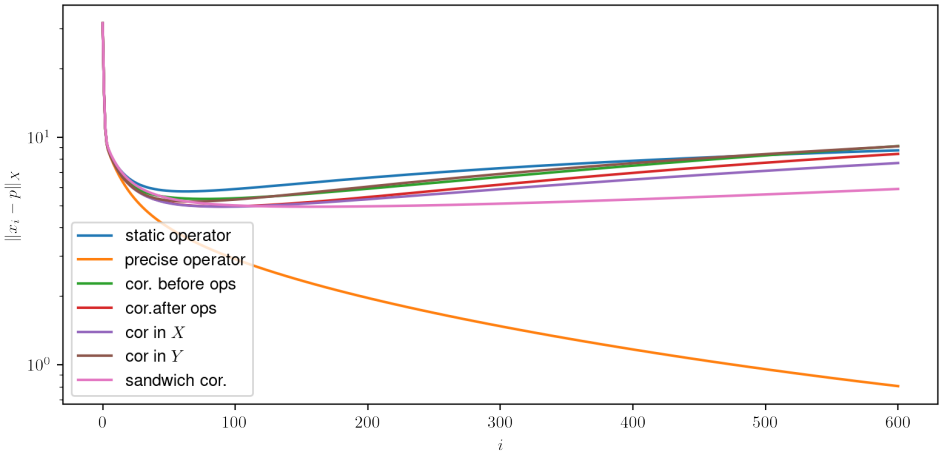

Figure 13: reconstruction with corrected operator and ISTA with a shift the model is trained on and the too large $\mu$ it is trained on

after that is the one with corrections only in the phantom space, and the one after that is the one used by Lunz et al. [15]. Both have a correction after $\tilde{A}^*$ – this is probably beneficial, because it gives a correction at the end of the computation of $F_\Theta^\dagger(x)$. That the correction operating only in the phantom space is better than the one operating in both phantom and data spaces is surprising.

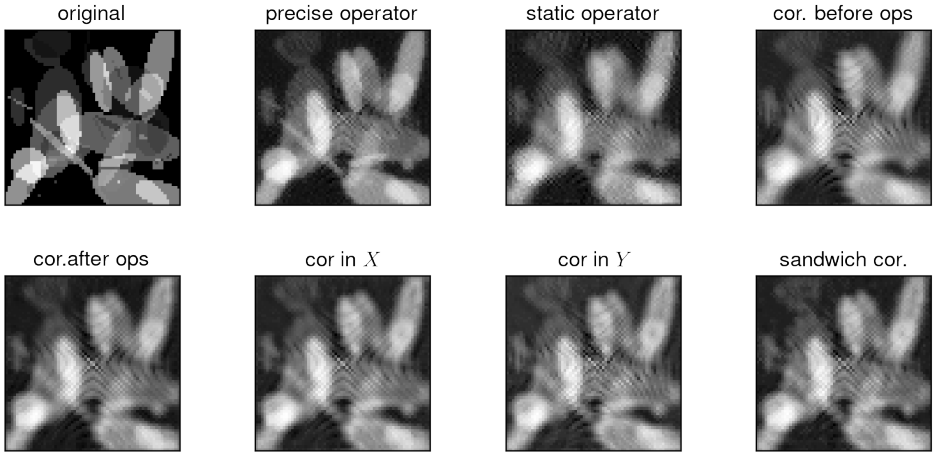
In Figure 15 we see the condition we put on the training losses, the y-axis is scaled logarithmic so we can better see, when the function fall under 0 and our conditions are no longer fulfiled. We can see, that the condition on our training losses are in this case stricter then they have to be. This might mean that the inequalities we used in the proof are not that strict, and that there is room for improvement.

We will now look at how well the corrections generalize for different shifts. In Figure 16a we see that if we use a shift which the corrections were not trained on, the parameters of the shift are still the same. We see that at the beginning the reconstruction, with the corrections in the positions 1,3 and 5, are still better then no correction. But only the loss of the correction, which sandwiches the operators, stays below the loss of the uncorrected reconstruction.

The parameters of the shift used in Figure 16b are still similar to those of the training shifts, with the difference that the amplitude is larger ($a_{u,i} = a_{v,i} = 0.15, b_{u,i} = b_{v,i} = 1000, c_{u,i} = c_{v,i} = 0, n = 1$ for both the $u$ and $v$ shift). Here we see again that the three best positionings of the correction are in positionings 1, 3 and 5. While their loss does not increase as fast as the static operator, their loss is also always larger than the minimum loss of the static operator.




(a) loss for the different trained corrections




(b) reconstrutions for $i = 49$

Figure 14: reconstruction with corrected operator and ISTA with a shift the model is trained on, with $\mu = 0.002$

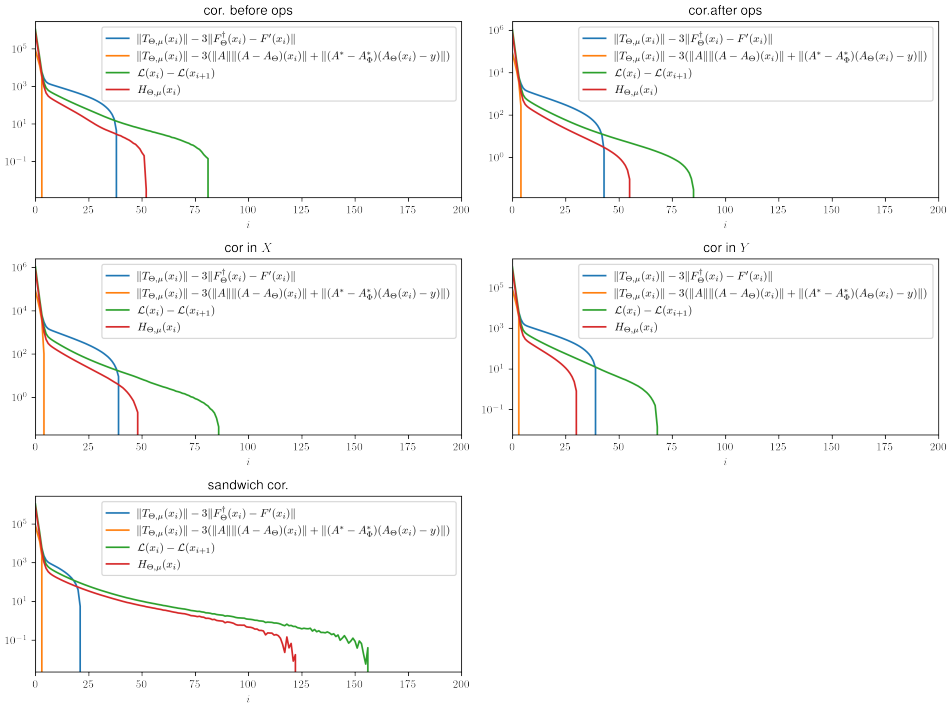
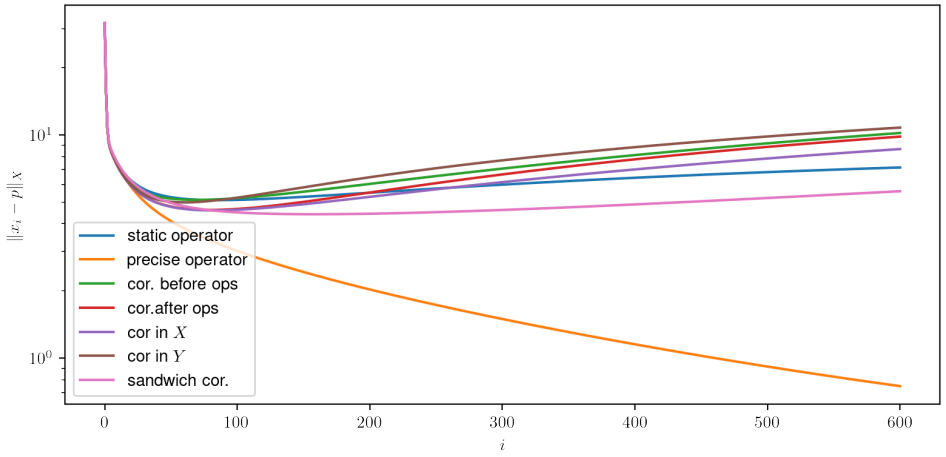



Figure 15: performance of our theoretical bounds

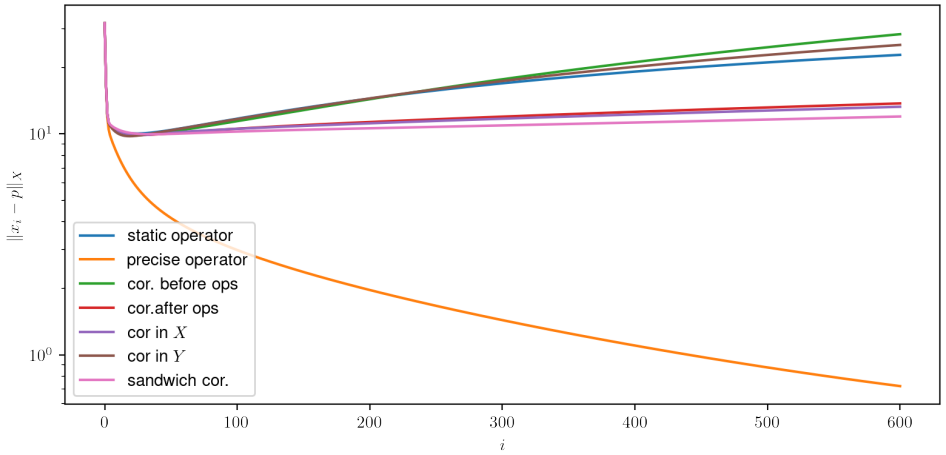
Finally, the shift used in Figure 16c has again parameters $a_{u,i}, a_{u,i} \in [0.03, 0.05], b_{u,i}, b_{u,i} \in [500, 5000], c_{u,i}, c_{u,i} \in [0, 2\pi]$ but now with $n = 5$. In this case, we see that the corrections are, apart from the very early iterations, worse than the corrected operator.

In conclusion we can say that the corrections need to be trained on shifts similar to those they will be deployed on after training. This was to be expected.

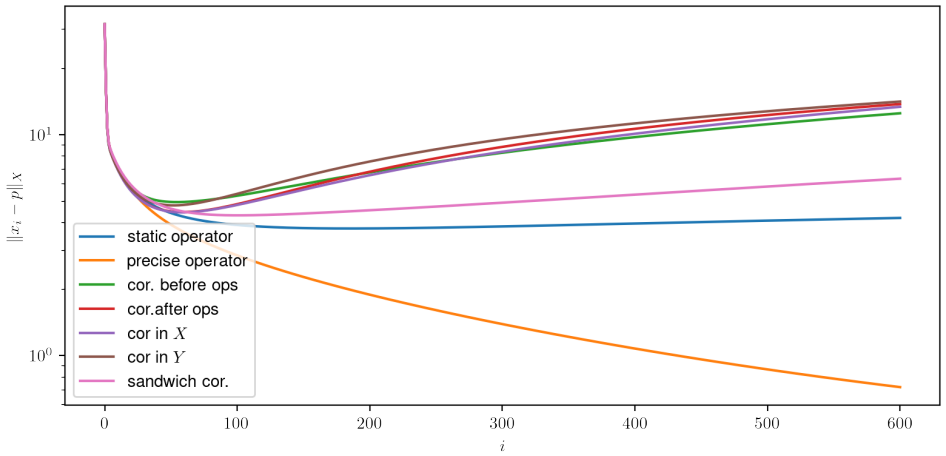
*Remark* 4.2. For the training I used a server with a NVIDIA A30 with 24GB and 24 CPU cores (AMD EPYC 7443P) provided by the University of Hamburg. The training for the 50 iterations of the corrections with one U-net for $\tilde{A}$ and $\tilde{A}^*$ with the maximum training sets size of 5000 took about 21–23 hours each. Training the sandwich correction for 100 itinerations with a maximum training set size of 2000 took 33 hours.




(a) similar shift that in training




(b) shift with larger amplitude




(c) shift with $n = 5$ sinus waves

Figure 16: loss for shifts that where not used in training

# 5 Conclusion

In this Master's thesis we have seen a theoretical result which gives us for a neighbourhood $U$ around the solutions of the objective function, with the precise operator, an upper bound on the difference

$$\|A^*(Ax - y) - A_\Phi^*(A_\Theta(x) - y)\|,$$

for corrected operator $A_\Theta$ and the precise operator $A$. If this upper bound holds for all $x \notin U$, we get the guarantees that the proximal gradient method, using $A_\Phi^*(A_\Theta(x) - y)$, will reach $U$.

For the theory to work, $A_\Theta$ and $A_\Phi^*$ do not need to be trained corrections, i.e. the findings presented here provide some groundwork for other correction approaches as well. We could also use the functions we derived in a different way, for example applying them to neural nets which are not dependent on $\tilde{A}$ or to a correction that is built with additional knowledge.

The numerical examples of corrections presented in this Master's thesis show that it is possible to train a correction operator for a Radon transform, with missing information about shifts of the phantom, and get better results than without correcting. However, there is still a lot of room for improvement and possible choices of which neural network we use for the correction.

# References


[1] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[2] Amir Beck. *First-Order Methods in Optimization.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[3] Matthias Beckmann. Computer tomography. lecture notes of WS19/20, Universität Hamburg, 2020.

[4] Anna Christina Brandt. Inverse problems. lecture notes of WS21/22, Universität Hamburg, DE, 2022.

[5] Anna Christina Brandt, Armin Iske, and Tobias Kluth. Machine learning. lecture notes of SS21, Universität Hamburg, DE, 2021.

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 2:303–314.

[7] Olaf Dössel. Bildgebung in der medizin. In *Bildgebende Verfahren in der Medizin.* Springer, 2016.

[8] Hadi Fayad, Frederic Lamare, Thibaut Merlin, and Dimitris Visvikis. Motion correction using anatomical information in pet/ct and pet/mr hybrid imaging. *The quarterly journal of nuclear medicine and molecular imaging : official publication of the Italian Association of Nuclear Medicine (AIMN) [and] the International Association of Radiopharmacology (IAR), [and] Section of the Society of..*, 60(1):12—24, March 2016.

[9] Catherine F. Higham and Desmond J. Higham. Deep learning: An introduction for applied mathematicians. *SIAM Review*, 61(4):860–891, 2019.

[10] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[11] Kyu Gyeom Kim, Jae Hee Kim, Jong Hyun Ryu, Jong Hwan Min, Kyong Woo Kim, and Kwon-Ha Yoon. Nano x-ray computed tomography system. In R. Magjarevic and J. H. Nagel, editors, *World Congress on Medical Physics and Biomedical Engineering 2006*, pages 1417–1420, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Hang-Chin Lai and Lai-Jiu Lin. Moreau-rockafellar type theorem for convex set functions. *Journal of Mathematical Analysis and Applications*, 132(2):558–571, 1988.

[14] Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. NETT: solving inverse problems with deep neural networks. *Inverse Problems*, 36(6):065005, jun 2020.

[15] Sebastian Lunz, Andreas Hauptmann, Tanja Tarvainen, Carola-Bibiane Schonlieb, and Simon Arridge. On learned operator correction in inverse problems. *SIAM Journal on Imaging Sciences*, 14(1):92–127, 2021.

[16] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. Adversarial regularizers in inverse problems. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[17] Tim Meinhardt, Michael Moller, Caner Hazirbas, and Daniel Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[18] Frank Natterer. *The mathematics of computerized tomography.* SIAM, 2001.

[19] M. Prummer, L. Wigstrom, J. Hornegger, J. Boese, G. Lauritsch, N. Strobel, and R. Fahrig. Cardiac c-arm ct: Efficient motion correction for 4d-fbp. In *2006 IEEE Nuclear Science Symposium Conference Record*, volume 4, pages 2620–2628, 2006.

[20] Andreas Rieder. *Keine Probleme mit Inversen Problemen.* Springer, 2003.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[22] Alexander Sasov, Xuan Liu, and Phil L. Salmon. Compensation of mechanical inaccuracies in micro-CT and nano-CT. In Stuart R. Stock, editor, *Developments in X-Ray Tomography VI*, volume 7078, page 70781C. International Society for Optics and Photonics, SPIE, 2008.

# Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Mathematics selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

| | |
|---|---|
| Ort, Datum | Björn Ehlers |