Abstract

In this work we consider methods for solving large scale optimization problems with a possibly nonsmooth objective. The key idea is to first specify a class of optimization algorithms using a generic iterative scheme involving only linear operations and applications of proximal operators. This generic scheme contains many modern primal-dual first-order solvers like the Douglas–Rachford and hybrid gradient methods as special cases. Moreover, stability of this scheme is investigated and its performance is compared to state-of-the-art solvers. Next, we interpret the generic scheme as a recurrent neural network and use unsupervised training to learn the best set of parameters for a specific class of objective functions while imposing a fixed number of iterations. We demonstrate this principle of “learning to optimize” for the family of total variation regularization problems, where the entire procedure can be interpreted as learning a generalized inverse for an ill-posed inverse problem. As use cases, we consider optimization problems arising in tomographic reconstruction and image deconvolution with sparsity type of regularization.

1 Introduction

Many problems in science and engineering result in the need for solving an optimization problem of the form

$$\min_{x \in X} \left[ F(x) + \sum_{i=1}^{m} G_i(L_i x) \right].$$

Here, $L_i: \mathcal{X} \to \mathcal{Y}_i$, $i = 1, \ldots, m$, are linear operators, where $\mathcal{X}, \mathcal{Y}_1, \ldots, \mathcal{Y}_m$ are Hilbert spaces, and $F: \mathcal{X} \to \mathbb{R}$ and $G_i: \mathcal{Y}_i \to \mathbb{R}$, $i = 1, \ldots, m$, are proper, convex and lower semicontinuous functions.

As we briefly outline next, optimization problems of the type in (1.1) arise when solving ill-posed inverse problems by means of variational regularization.

*equal contribution
Inverse problems  The goal in an inverse problem is to recover parameters characterizing a system under investigation from indirect observations. This can be formalized as the task of estimating (reconstructing) model parameters, henceforth called signal, \( x_{\text{true}} \in \mathcal{X} \) from indirect observations (data) \( b \in \mathcal{Y} \) where

\[
b = T(x_{\text{true}}) + \delta b.
\]  

In the above, \( \mathcal{X} \) and \( \mathcal{Y} \) are typically Hilbert or Banach spaces, and \( T: \mathcal{X} \to \mathcal{Y} \) (forward operator) models how a given signal gives rise to data in absence of noise. Furthermore, \( \delta b \in \mathcal{Y} \) is a single sample of a \( \mathcal{Y} \)-valued random element that represents the noise component of data.

Many of the advanced imaging technologies gives rise to such inverse problems, e.g., X-ray computed tomography (CT) \cite{36,37}, magnetic resonance imaging (MRI) \cite{16}, and electron tomography \cite{38}.

A natural approach for solving (1.2) is to minimize a function \( R: \mathcal{X} \to \mathbb{R} \) (data discrepancy functional) that quantifies the miss-fit in data space. This function needs to incorporate the aforementioned forward operator \( T \) and the data \( b \), it is often of the form

\[
R(x) := \ell(T(x), b) \quad \text{for some } \ell: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}.
\]

If \( \ell \) is the negative data log-likelihood, then minimizing \( x \mapsto R(x) \) corresponds to finding a maximum likelihood solution to (1.2).

However, only minimizing \( R \) is an ill-posed problem, meaning that a solution (if it exists) is discontinuous with respect to the data \( b \). Variational regularization addresses this issue by introducing an additional function \( S: \mathcal{X} \to \mathbb{R}_+ \) (regularization functional) that encodes a priori information about \( x_{\text{true}} \) and penalizes undesirable solutions \cite{24}. This results in an optimization problem

\[
\min_{x \in \mathcal{X}} \left[ \lambda R(x) + S(x) \right],
\]

which from a statistical perspective can be interpreted as trying to find a maximum a posteriori estimate \cite{27}. A common choice of regularization functional, especially for inverse problems in imaging, is the total variation (TV) regularization \( S(x) := \| \nabla x \|_1 \), but several more advanced regularizers have also been suggested in the literature, typically exploiting some kind of sparsity using a \( L_1 \)-like norm \cite{17}.

A key challenge is to handle the computational burden. In imaging, and especially so for three-dimensional imaging, the resulting optimization problem is very high-dimensional even after clever digitization and might involve more than one billion variables. Furthermore, computing \( x \mapsto T(x) \) and the adjoint of its derivative is very costly. Finally, many regularizers that are popular in imaging, like those associated with sparsity, result in a nonsmooth objective function. These issues prevent usage of variational methods in time-critical applications, such as medical imaging in a clinical setting.

2 Outline of the paper

The main objective of the paper is to offer a computationally feasible approach for minimizing large-scale nondifferentiable, convex functions. The key idea is to
“learn” how to optimize from training data, resulting in an iterative scheme that is optimal given a fixed number of steps, while its convergence properties can be analyzed. This has been done previously in [7, 25, 32], but these approaches are either limited to specific classes of iterative schemes, like gradient-descent-like schemes [7, 32] that are not applicable for nonsmooth optimization, or specialized to a specific class of regularizers as in [25], which limits the possible choices of regularizers and forward operators.

The approach taken here leverages upon these ideas and yields a general framework for learning optimization algorithms that are applicable to solving optimization problems of the type (1.1). Our framework is inspired by modern methods for nonsmooth convex optimization that are typically based on the proximal point algorithm [41] and operator splitting techniques, see e.g., [9, 11, 13–15, 18–20, 23, 26, 30, 31] and references therein.

A key feature is to present a general formulation that includes several existing algorithms, among them the primal-dual hybrid gradient (PDHG) algorithm (also called the Chambolle–Pock algorithm) [18] and the primal-dual Douglas–Rachford algorithm [14] as a special case. We demonstrate the performance of a solver based on this general formulation by training in an unsupervised manner for two inverse problems: image reconstruction in CT and deconvolution, both through TV regularization. Moreover, empirical results indicate that by adding additional memory we can achieve improved performance.

3 Background

Solving optimization problems of the type in (1.1) are often addressed using variable splitting techniques, which work well if the different terms are “simple” [9, 20, 22]. The following subsections present necessary background material on monotone operators, convex optimization, and variable splitting.

3.1 Fundamental notions

Let $H$ be a real Hilbert space with the inner product $\langle \cdot, \cdot \rangle$. We denote convergence in norm (or strong convergence) and weak convergence by $\to$ and $\rightharpoonup$, respectively. A set-valued operator $S: H \rightharpoonup H$ is monotone if

$$\langle z - z', w - w' \rangle \geq 0 \quad \text{for all } z, z' \in H \text{ with } w \in S(z) \text{ and } w' \in S(z'),$$

and it is called maximally monotone if, in addition, the graph of $S$, defined by $\text{graph}(S) := \{(z, w) \in H \times H \mid w \in S(z)\}$, is not properly contained in the graph of any other monotone operator. This means that

$$(z, w) \in \text{graph}(S) \iff \langle z - z', w - w' \rangle \geq 0 \text{ for all } (z', w') \in \text{graph}(S).$$

A monotone operator is called strongly monotone if there exists a $\mu > 0$ such that

$$\langle z - z', w - w' \rangle \geq \mu \|z - z'\|^2 \quad \text{for all } z, z' \in H \text{ with } w \in S(z) \text{ and } w' \in S(z').$$

Next, for any scalar $\sigma > 0$, the operator $(\text{Id} + \sigma S)^{-1}$ is called the resolvent operator or proximal mapping [41].
A **maximally monotone inclusion problem** is now defined as the problem of finding a point \( z \in H \) such that \( 0 \in S(z) \), which we henceforth denote \( z \in \text{zer}(S) \). In fact, it is easily seen that \( z \in \text{zer}(S) \) is equivalent with \( z \) being a fixed-point for the resolvent operator, i.e., \( z = (\text{Id} + \sigma S)^{-1}(z) \). One reason for the interest in maximally monotone inclusion problems is that the subdifferential of a proper, convex and lower semicontinuous function on \( F: H \to \mathbb{R} \) is maximally monotone operator \([35]\). Moreover, the subdifferential at any global minimizer of such a function contains zero, so the convex minimization of \( F \) can be solved by solving a maximally monotone inclusion problem.

The **proximal point algorithm** is a fixed-point iterative scheme for solving the maximally monotone inclusion problem. It is given by repeatedly applying the resolvent operator:

\[
z_{k+1} = (\text{Id} + \sigma S)^{-1}(z_k).
\]

It can now be shown that if \( \text{zer}(S) \neq \emptyset \) then \( z^k \) converges weakly to a point \( z^\infty \in \text{zer}(S) \) \([41]\). The special case when \( S := \partial F \), i.e., the case of the resolvent of a subdifferential of \( F \), is called the **proximal operator**. One can express the proximal as \([35]\)

\[
(\text{Id} + \sigma \partial F)^{-1}(z) = \text{Prox}_\sigma F(z) = \arg \min_{z'} \left\{ F(z') + \frac{1}{2\sigma} \| z' - z \|^2 \right\}.
\]

(3.1)

To see this, we simply note that if \( z' \) is a minimizing argument then

\[
0 \in \partial F(z') + \frac{1}{\sigma} (z' - z) \iff z' = (\text{Id} + \sigma \partial F)^{-1}(z).
\]

It is thus interesting to note that the fixed-point iteration

\[
z_{k+1} = \text{Prox}_\sigma F(z_k) = \arg \min_{z'} \left\{ F(z') + \frac{1}{2\sigma} \| z' - z_k \|^2 \right\}
\]

generates a sequence \( (z_k) \) that converges weakly to a global minimizer of \( F \). In this setting, the parameter \( \sigma \) can be interpreted as a step length. This can give rise to methods for solving the optimization problems if the proximal operator can be efficiently computed, e.g., through a closed-form expression.

Sometimes the resolvent of the maximally monotone operator \( S \) is not easy to evaluate, but \( S \) is of the form \( S = A + B \) where \( A \) and \( B \) are maximally monotone and the resolvents of \( A \) and \( B \) can be evaluated efficiently. One may then consider approximating \( (I + \sigma (A + B))^{-1} \) with \( (I + \sigma A)^{-1} \) and \( (I + \sigma B)^{-1} \) (splitting) \([22]\). An example when this arises is in convex minimization of an objective that is a sum of two (or more) functions \( F + G \), like in (1.1). In these cases it is often not possible to compute a closed-form expression for the proximal operator \( \text{Prox}_{\sigma F + G} \). Such problems can be addressed using operator splitting techniques that allow for solving the problem by only evaluating \( \text{Prox}_\sigma F \) and \( \text{Prox}_\sigma G \) \([20]\).

### 3.2 Convex optimization

To simplify the notation, we consider the case \( m = 1 \) in (1.1), i.e., let \( X \) and \( Y \) be two Hilbert spaces and consider the model problem

\[
\min_{x \in X} \left\{ F(x) + G(Lx) \right\},
\]

(3.2)

*In optimization, this operator splitting is sometimes referred to as *variable splitting*. The reason for this can be understood by comparing equations (3.2) and (3.6) below.*
where $L: X \to Y$ is a continuous linear operator and $F: X \to \mathbb{R}$ and $G: Y \to \mathbb{R}$ are proper, convex and lower semicontinuous functions. Note that (1.1) is recovered by setting $G(y) := \sum_{i=1}^{m} G_i(y_i)$ for $y = (y_1, \ldots, y_m) \in Y := Y_1 \times \cdots \times Y_m$ and $Lx := (L_1x, \ldots, L_mx)$ for $x \in X$ in (3.2).

The dual formulation of the primal problem (3.2) is
\begin{equation}
\max_{y \in Y} \left[ -F^*(L^*y) - G^*(-y) \right], \tag{3.3}
\end{equation}
where $F^*$ is the Fenchel dual (convex conjugate) function \cite[Chapter 13]{9}. Under suitable conditions the two optimization problems (3.2) and (3.3) have the same optimal value \cite[Chapter 15.3]{9}. Also note that since both $F$ and $G$ are proper, convex and lower semicontinuous functions, $F^{**} = F$ and $G^{**} = G$ by the Fenchel–Moreau theorem \cite[Theorem 13.37]{9}. Hence, the following primal-dual formulation
\begin{equation}
\min_{x \in X} \max_{y \in Y} \mathcal{L}(x; y) \quad \text{with} \quad \mathcal{L}(x; y) = \langle Lx, y \rangle + F(x) - G^*(y) \tag{3.4}
\end{equation}
is equivalent to the primal problem.\footnote{To see this, note that $\max_{y \in Y} [\langle Lx, y \rangle - G^*(y)] = G^{**}(Lx) = G(Lx).$} In fact, under suitable assumptions it can be shown that if $(\bar{x}, \bar{y})$ is a saddle point to (3.4), then $\bar{x}$ is a solution to the primal problem (3.2) and $\bar{y}$ is a solution to the dual problem (3.3) \cite[Proposition 19.20]{9}.

A necessary optimality condition for the primal-dual formulation (3.4) is that the corresponding point be stationarity with respect to both variables, i.e., that
\begin{equation}
Lx \in \partial G^*(y) \quad \text{and} \quad -L^*y \in \partial F(x). \tag{3.5}
\end{equation}
For later use we note that the first of these conditions can be reformulated as
\begin{align*}
Lx \in \partial G^*(y) & \iff y + \sigma Lx \in y + \sigma \partial G^*(y) = (I + \sigma \partial G^*)(y) \\
& \iff y = (I + \sigma \partial G^*)^{-1}(y + \sigma Lx) = \text{Prox}_{\sigma G^*}(y + \sigma Lx),
\end{align*}
and the second as
\begin{align*}
-L^*y \in \partial F(x) & \iff x - \tau L^*y \in x + \tau \partial F(x) = (I + \tau \partial F)(x) \\
& \iff x = (I + \tau \partial F)^{-1}(x - \tau L^*y) = \text{Prox}_{\tau F}(x - \tau L^*y).
\end{align*}
Therefore, an equivalent condition to (3.5) is
\begin{equation}
y = \text{Prox}_{\sigma G^*}(y + \sigma Lx) \quad \text{and} \quad x = \text{Prox}_{\tau F}(x - \tau L^*y). \tag{3.6}
\end{equation}
3.3 Two splitting algorithms

As mentioned before, there are many different splitting methods available to solve problems of the form (1.1). We here mention two popular choices. The first one, given in (3.7), is PDHG [18].

\[
\begin{align*}
v_{n+1} &= \text{Prox}_{G^*}(v_n + \sigma L y_n), \\
x_{n+1} &= \text{Prox}_{F^*}(x_n - \tau L^* v_n) \\
y_{n+1} &= x_{n+1} + \theta(x_{n+1} - x_n)
\end{align*}
\]

(3.7)

The second one is the Douglas–Rachford type primal-dual algorithm [14], presented in (3.8).

\[
\begin{align*}
p_{1,n} &= \text{Prox}_{F^*}(x_n - \tau L^* v_n) \\
x_{n+1} &= x_n + \lambda_n(p_{1,n} - x_n) \\
p_{2,n} &= \text{Prox}_{G^*}(v_n + \sigma L (2p_{1,n} - x_n)), \\
v_{n+1} &= v_n + \lambda_n(p_{2,n} - v_n)
\end{align*}
\]

(3.8)

Remark 3.1. Note that an efficient routine to evaluate \(\text{Prox}_F(x)\) yields an efficient routine to evaluate \(\text{Prox}_{F^*}(x)\), and vice versa. This follows from the Moreau decomposition that allows one to relate these two by \(\text{Prox}_{\tau F}(x) = x - \tau \text{Prox}_{1/\tau F^*}(x/\tau)\) [9, Theorem 14.3].

4 A new family of optimization solvers

In this section we introduce a new family of optimization algorithms and prove convergence for a subfamily. For ease of notation we will consider the simplified optimization problem (3.2), but results easily extend to the general case (1.1). To this end, considering the two algorithms (3.7) and (3.8). Note that that they can both be written as

\[
\begin{align*}
q_n &= \text{Prox}_{G^*}(b_{12} y_n + b_{11} L(a_{11} p_{n-1} + a_{12} x_{n-1})), \\
y_{n+1} &= c_{21} q_n + c_{22} y_n, \\
p_n &= \text{Prox}_{F^*}(d_{12} x_n + d_{11} L^* (c_{11} q_n + c_{12} y_n)), \\
x_{n+1} &= a_{21} p_n + a_{22} x_n,
\end{align*}
\]

(4.1a)

for suitable values of the coefficients. More precisely, the PDHG algorithm (3.7) is obtained by setting

\[
\begin{align*}
a_{11} &= 1 + \theta & a_{12} &= -\theta & a_{21} &= 1 & a_{22} &= 0 & b_{12} &= \sigma & b_{12} &= 1 \\
c_{11} &= 1 & c_{12} &= 0 & c_{21} &= 1 & c_{22} &= 0 & d_{11} &= -\tau & d_{12} &= 1
\end{align*}
\]

and the Douglas-Rachford algorithm (3.8) by setting

\[
\begin{align*}
a_{11} &= 2 & a_{12} &= -1 & a_{21} &= \lambda_n & a_{22} &= 1 - \lambda_n & b_{12} &= \sigma & b_{12} &= 1 \\
c_{11} &= \lambda_n & c_{12} &= 1 - \lambda_n & c_{21} &= \lambda_n & c_{22} &= 1 - \lambda_n & d_{11} &= -\tau & d_{12} &= 1
\end{align*}
\]

We now go on to analyze the scheme (4.1). To state our results as generally as possible, we formulate them for a monotone inclusion problem that in
particular specializes to the optimality conditions in (3.6) when the operators are subdifferentials. The monotone inclusion problem we seek to solve reads as follows: Let \( X \) and \( Y \) be two (not necessarily finite-dimensional) Hilbert spaces, and let \( L : X \to Y \) be a continuous linear operator. Let \( A : X \rightrightarrows X \) and \( B : Y \rightrightarrows Y \) be maximally monotone operators. Find a pair \((\bar{x}, \bar{y})\) \(\in X \times Y\) such that

\[
L \bar{x} \in B^{-1} \bar{y} \quad \text{and} \quad -L^* \bar{y} \in A \bar{x}.
\]

(4.2)

Remark 4.1. A continuous linear operator \( A : H \to H \) of a Hilbert space \( H \) into itself is maximally monotone if and only if it is accretive, i.e., if \( \langle x, Ax \rangle \geq 0 \) for all \( x \in H \) [9, Corollary 20.28, see also Definition 2.23], and it is the subdifferential \( \partial f \) of a function \( f : H \to \mathbb{R} \) if and only if it is additionally symmetric [8, Proposition 2.51]. In particular the Volterra integral operator [10, Example 4.4]

\[
(Af)(t) = \int_0^t f(s) \, ds
\]

and its inverse are maximally monotone, but not the subdifferential of a proper, convex and lower semicontinuous function.

In this setting, the scheme (4.1) generalizes to

\[
q_n = J_{\sigma B^{-1}}(b_{12}y_n + b_{11}L(a_{11}p_{n-1} + a_{12}x_{n-1})),
\]

(4.3a)

\[
y_{n+1} = c_{21}q_n + c_{22}y_n,
\]

(4.3b)

\[
p_n = J_{\tau A}(d_{12}x_n + d_{11}L^*(c_{11}q_n + c_{12}y_n)),
\]

(4.3c)

\[
x_{n+1} = a_{21}p_n + a_{22}x_n.
\]

(4.3d)

We first note that if \( c_{21} = 0 \) or \( a_{21} = 0 \) the update for either \( y_{n+1} \) or \( x_{n+1} \) becomes trivial, and the algorithm cannot be globally convergent to an optimal solution to (3.2). Henceforth we will therefore assume that \( c_{21} \) and \( a_{21} \) are not equal to 0, unless the opposite is explicitly stated.

4.1 Fixed-point analysis

In this section, we give sufficient conditions for the solution set of (4.2) and the fixed point set of (4.3) to coincide. Let \((\bar{q}, \bar{y}, \bar{p}, \bar{x}) \in \mathcal{Y} \times \mathcal{Y} \times \mathcal{X} \times \mathcal{X}\) be a fixed point of the iterative scheme (4.3). Our aim is to formulate necessary and sufficient conditions for the scheme so that \((\bar{x}, \bar{y})\) satisfies (4.2) for any choice of \( A, B \) and \( L \). We start by defining

\[
\bar{q} := \frac{1 - c_{22}}{c_{21}} \bar{y} \quad \text{and} \quad \bar{p} := \frac{1 - a_{22}}{a_{21}} \bar{x}.
\]

Using this notation, we further get that

\[
\frac{1 - c_{22}}{c_{21}} \bar{y} = J_{\sigma B^{-1}}(b_{12} \bar{y} + b_{11}L\left(\frac{1 - a_{22}}{a_{21}} \bar{x} + a_{12} \bar{x}\right))
\]

\[
\frac{1 - a_{22}}{a_{21}} \bar{x} = J_{\tau A}(d_{12} \bar{x} + d_{11}L^*\left(\frac{1 - c_{22}}{c_{21}} \bar{y} + c_{12} \bar{y}\right))
\]

The conditions in (4.2) can now be re-phrased as

\[
\bar{y} = J_{\sigma B^{-1}}(\bar{y} + \sigma L \bar{x}) \quad \text{and} \quad \bar{x} = J_{\tau A}(\bar{x} - \tau L^* \bar{y}),
\]
and combining the above two equations yields

\[ c_{21} + c_{22} = 1, \quad b_{12} = 1, \quad b_{11}(a_{11} + a_{12}) = \sigma, \]
\[ a_{21} + a_{22} = 1, \quad d_{12} = 1, \quad d_{11}(c_{11} + c_{12}) = -\tau. \]  
(4.4)

Observe that, due to the linearity of \( L \), the algorithm does not change if we agree to the normalization

\[ b_{11} = \sigma, \quad a_{11} + a_{12} = 1, \quad d_{11} = -\tau, \quad c_{11} + c_{12} = 1. \]

If we fix all these conditions, the iteration (4.1) takes the form

\[ q_n = J_{\sigma B^{-1}}(y_n + \sigma L(x_{n-1} + a_{11}(p_{n-1} - x_{n-1}))), \]
(4.5a)
\[ y_{n+1} = y_n + c_{21}(q_n - y_n), \]
(4.5b)
\[ p_n = J_{\tau A}(x_n - \tau L^*(y_n + c_{11}(q_n - y_n))), \]
(4.5c)
\[ x_{n+1} = x_n + a_{21}(p_n - x_n). \]
(4.5d)

### 4.2 Convergence analysis

The following theorem gives sufficient conditions for the weak convergence of the sequence \((x_n, y_n)\) generated by (4.5) to a point that satisfies (4.2), i.e., a point that solves the monotone inclusion problem.

**Theorem 4.2.** Assume that there is a point that satisfies (4.2), i.e., the monotone inclusion problem has a solution. Moreover, let

\[ c_{11} = c_{21} \quad \text{and} \quad a_{11} = 1 + \frac{a_{21}}{c_{21}}, \]  
(4.6)

Assume furthermore that \( 0 < a_{21} < 2 \) and \( 0 < c_{21} < 2 \) and

\[ \sigma \tau \|L\|^2 < \frac{c_{21}^2(2 - a_{21})(2 - c_{21})}{(a_{21} + c_{21} - a_{21}c_{21})^2} \]  
with \( \sigma, \tau > 0. \)  
(4.7)

Finally, let \((q_n, y_n, p_n, x_n)\) be the sequence generated by scheme (4.5). Then the following holds:

(a) \( \sum_{n \geq 0} \|x_n - p_n\|^2 < +\infty \) and \( \sum_{n \geq 0} \|x_n - x_{n+1}\|^2 < +\infty, \)

(b) \( \sum_{n \geq 0} \|y_n - q_n\|^2 < +\infty \) and \( \sum_{n \geq 0} \|y_n - y_{n+1}\|^2 < +\infty, \)

(c) The sequence \((x_n, y_n)\) converges weakly to a point that satisfies (4.2).

(d) If \( A \) is strongly monotone, then there is a unique \( \bar{x} \in X \) such that all solutions of (4.2) are of the form \((\bar{x}, y)\) with some \( y \in Y \). Moreover,

\[ \sum_{n=1}^{\infty} \|p_n - \bar{x}\|^2 < +\infty, \]  
in particular \( p_n \to \bar{x} \) strongly.

If \( B^{-1} \) is strongly monotone, then there is a unique \( \bar{y} \in Y \) such that all solutions of (4.2) are of the form \((x, \bar{y})\) with some \( x \in X \). Moreover,

\[ \sum_{n=1}^{\infty} \|q_{n+1} - \bar{y}\|^2 < +\infty, \]  
in particular \( q_n \to \bar{y} \) strongly.
Remark 4.3. By rewriting with (4.6), the iteration (4.5) becomes

\[ q_n = J_{\sigma B} - 1 \left( y_n + \sigma L \left( p_{n-1} + \frac{a_{21}}{c_{21}} (p_{n-1} - x_{n-1}) \right) \right), \]  
(4.8a)

\[ y_{n+1} = y_n + c_{21} (q_n - y_n), \]  
(4.8b)

\[ p_n = J_{\tau A} (x_n - \tau L^* y_{n+1}), \]  
(4.8c)

\[ x_{n+1} = x_n + a_{21} (p_n - x_n). \]  
(4.8d)

The remainder of the convergence analysis will therefore refer to scheme (4.8).

The proof of the theorem rests upon a number of technical results and is given in Section 4.2.1. An immediate corollary is the convergence of the primal-dual Douglas–Rachford method with constant relaxation [14].

Corollary 4.4. Let \( \sigma \|L\| < 1 \) and \( 0 < \lambda < 2 \). Then, for the iteration

\[ q_n = J_{\sigma B} - 1 \left( y_n + \sigma L (2 p_{n-1} - x_{n-1}) \right), \]
\[ y_{n+1} = y_n + \lambda (q_n - y_n), \]
\[ p_n = J_{\tau A} (x_n - \tau L^* y_{n+1}), \]
\[ x_{n+1} = x_n + \lambda (p_n - x_n), \]

the sequence \((x_n, y_n)_n\) converges weakly to a point that satisfies (4.2).

Proof. Set \( \lambda = a_{21} = c_{21} \) in Theorem 4.2 and observe that (4.7) reduces to \( \sigma \|L\|^2 < 1 \).

4.2.1 Proof of Theorem 4.2

For the proof, we define notions of distance \( Q_1 \) and \( Q_2 \) on the space \( \mathcal{X} \times \mathcal{Y} \) of pairs of primal and dual variables (Lemma 4.6). Next, we show that the distance (in terms of \( Q_1 \)) between the iterates and the set of solutions of (4.2) decreases (Proposition 4.7). This property is also known as Fejér monotonicity [9, Chapter 5]. Proposition 4.8 improves the statement of Proposition 4.7 for strongly monotone operators. The proof of Theorem 4.2 is completed by showing that any weak sequential cluster point of the iteration sequence is a solution to (4.2).

We start with some simple inequalities between real numbers. In particular, Lemma 4.5 shows that we do not divide by zero in (4.7).

Lemma 4.5. Let \( 0 < a_{21} < 2 \) and \( 0 < c_{21} < 2 \). Then

(a) \( a_{21} + c_{21} > a_{21} c_{21} \) and

(b) \( \frac{a_{21} c_{21} (2 - a_{21}) (2 - c_{21})}{(a_{21} + c_{21} - a_{21} c_{21})^2} \leq 1. \)

Proof. By assumption, \( a_{21} (2 - a_{21}) > 0 \), i.e., \( a_{21} > \frac{1}{2} a_{21}^2 \), and the same holds for \( c_{21} \). Therefore,

\[ a_{21} + c_{21} > \frac{1}{2} a_{21} + \frac{1}{2} c_{21} \geq a_{21} c_{21}, \]

whence (a).
For \([b]\), use the inequality \(2a_{21}c_1 \leq a_{21}^2 + c_1^2\) in
\[
a_{21}c_1(2 - a_{21})(2 - c_1) = 4a_{21}c_1 - 2a_{21}^2c_1 - 2a_{21}c_1^2 + a_{21}^2c_1^2
\leq a_{21}^2 + c_1^2 + 2a_{21}c_1 - 2a_{21}^2c_1 - 2a_{21}c_1^2 + a_{21}^2c_1^2
= (a_{21} + c_1 - a_{21}c_1)^2.
\]

**Lemma 4.6.** Define the quadratic forms \(Q_1, Q_2 : X \times \mathcal{Y} \to \mathbb{R}\) by
\[
Q_1(x, y) = \frac{1}{2\tau a_{21}} \|y - \sigma Lx\|^2 + \frac{1}{2\sigma c_1} \|x\|^2 - \frac{1}{c_1} \langle y, Lx \rangle,
\]
\[
Q_2(x, y) = \frac{2 - a_{21}}{2\tau} \|y - \frac{\sigma(a_{21} + c_1 - a_{21}c_1)}{c_1(2 - c_1)} Lx\|^2
+ \frac{2 - a_{21}}{2\tau} \|x\|^2 - \frac{\sigma(a_{21} + c_1 - a_{21}c_1)}{c_1(2 - c_1)} \|Lx\|^2,
\]
for all \(x \in X\) and \(y \in \mathcal{Y}\). Under the assumptions in Theorem 4.2, there exist \(C_1, C_2, D_1, D_2 > 0\) such that
\[
Q_1(x, y) \geq C_i \|x\|^2 \quad \text{and} \quad Q_i(x, y) \geq D_i \|y\|^2
\]
for all \(x \in X\), \(y \in \mathcal{Y}\) and \(i = 1, 2\).

**Proof.** We can rewrite
\[
Q_1(x, y) = \frac{1}{2\tau a_{21}} \|y - \sigma Lx\|^2 + \frac{1}{2\sigma c_1} \|x\|^2 - \frac{1}{c_1} \langle y, Lx \rangle,
\]
and
\[
Q_2(x, y) = \frac{2 - a_{21}}{2\tau} \|y - \frac{\sigma(a_{21} + c_1 - a_{21}c_1)}{c_1(2 - c_1)} Lx\|^2
+ \frac{2 - a_{21}}{2\tau} \|x\|^2 - \frac{\sigma(a_{21} + c_1 - a_{21}c_1)}{c_1(2 - c_1)} \|Lx\|^2.
\]
From this, the assertion of the theorem is clear with the quantities
\[
C_1 = \frac{1}{2\tau a_{21}} - \frac{\sigma}{2\sigma c_1} \|L\|^2 = \frac{c_1 - a_{21}\sigma L\|L\|^2}{2\tau a_{21}c_1},
\]
\[
D_1 = \frac{1}{2\tau a_{21}} - \frac{a_{21}\sigma}{2\sigma c_1} \|L\|^2 = \frac{c_1 - a_{21}\sigma L\|L\|^2}{2\sigma c_1},
\]
\[
C_2 = \frac{2 - a_{21}}{2\tau} - \frac{\sigma(a_{21} + c_1 - a_{21}c_1)^2}{2\sigma c_1^2(2 - c_1)} \|L\|^2
= \frac{c_1^2(2 - a_{21})(2 - c_1) - (a_{21} + c_1 - a_{21}c_1)^2\sigma \|L\|^2}{2\tau c_1^2(2 - c_1)},
\]
\[
D_2 = \frac{2 - a_{21}}{2\tau} - \frac{\sigma(a_{21} + c_1 - a_{21}c_1)^2}{2\sigma c_1^2(2 - a_{21})} \|L\|^2
= \frac{c_1^2(2 - a_{21})(2 - c_1) - (a_{21} + c_1 - a_{21}c_1)^2\sigma \|L\|^2}{2\sigma c_1^2(2 - c_1)}.
\]
provided that the numerators are positive, i.e.,
\[
\sigma r \| L \|^2 < \min \left\{ \frac{c_{21}}{a_{21}}, \frac{c_{21}^2 (2 - a_{21}) (2 - c_{21})}{a_{21} + c_{21} - a_{21} c_{21}} \right\}.
\]

Now, by Lemma 4.5, the minimum is always attained by the second value, and positivity is guaranteed by (4.7).

**Proposition 4.7.** Define \( Q_1 \) and \( Q_2 \) as in Lemma 4.6, let \((\bar{x}, \bar{y}) \in X \times Y\) satisfy (4.2), and let the sequence \((q_n, y_n, p_n, x_n)\) be generated by scheme (4.8). Under the assumptions in Theorem 4.2, we have for all \( n \geq 1 \)
\[
Q_1(x_n + 1 - \bar{x}, y_n + 2 - \bar{y}) - Q_1(x_n - \bar{x}, y_n + 1 - \bar{y}) \leq -Q_2(p_n - x_n, q_n + 1 - y_n + 1).
\]

**Proof.** Let \((\bar{x}, \bar{y})\) satisfy (4.2). Then
\[
Q_1(x_n + 1 - \bar{x}, y_n + 2 - \bar{y}) - Q_1(x_n - \bar{x}, y_n + 1 - \bar{y})
= \frac{1}{2\sigma a_{21}} \left( \| x_n + 1 - \bar{x} \|^2 - \| x_n - \bar{x} \|^2 \right)
+ \frac{1}{2\sigma c_{21}} \left( \| y_n + 2 - \bar{y} \|^2 - \| y_{n+1} - \bar{y} \|^2 \right)
+ \frac{1}{c_{21}} \left( \langle y_{n+1} - \bar{y}, L \bar{x} - y_n - L \bar{x} \rangle - \langle y_{n+2} - \bar{y}, L \bar{x} - y_n + 1 - L \bar{x} \rangle \right)
= \frac{1}{2\sigma a_{21}} \left( \| x_n - \bar{x} + a_{21}(p_n - x_n) \|^2 - \| x_n - \bar{x} \|^2 \right)
+ \frac{1}{2\sigma c_{21}} \left( \| y_n + 1 - \bar{y} + c_{21}(q_n + 1 - y_{n+1}) \|^2 - \| y_{n+1} - \bar{y} \|^2 \right)
+ \frac{1}{c_{21}} \left( \langle y_{n+1} - \bar{y} + c_{21}(q_n + 1 - y_{n+1}), L \bar{x} - y_n + 1 - L \bar{x} + a_{21}(L p_n - L x_n) \rangle \right)
= \frac{a_{21}}{2\sigma} \| p_n - x_n \|^2 + \frac{1}{\sigma} \langle x_n - \bar{x}, p_n - x_n \rangle + \frac{c_{21}}{2\sigma} \| q_n + 1 - y_{n+1} \|^2
+ \frac{1}{\sigma} \langle y_{n+1} - \bar{y}, q_{n+1} - y_{n+1} \rangle + \frac{a_{21}}{c_{21}} \langle \bar{y} - y_{n+1}, L p_n - L x_n \rangle
+ \langle q_{n+1} - y_{n+1}, L \bar{x} - L x_n \rangle + a_{21}(y_{n+1} - q_{n+1}, L p_n - L x_n). \tag{4.9}
\]

To estimate the above, we use the monotonicity of the operator \( B^{-1} \) together with the inclusions \( L \bar{x} \in B^{-1} \bar{y} \) from (4.2) and
\[
\bar{y} + \frac{q_{n+1} - q_{n+1}}{\sigma} + L p_n + \frac{a_{21}}{c_{21}} (L p_n - L x_n) \in B^{-1} q_{n+1}, \tag{4.10}
\]
which is a reformulation of (4.8) with \( n \) replaced by \( n + 1 \). This yields the inequality
\[
0 \leq \frac{\langle y_{n+1} - q_{n+1} \rangle}{\sigma} + L p_n + \frac{a_{21}}{c_{21}} (L p_n - L x_n) - L \bar{x}, q_{n+1} - \bar{y}
= \frac{1}{\sigma} \langle y_{n+1} - q_{n+1}, q_{n+1} - \bar{y} \rangle + \langle L p_n - L \bar{x}, q_{n+1} - \bar{y} \rangle
+ \frac{a_{21}}{c_{21}} (L p_n - L x_n, q_{n+1} - \bar{y}). \tag{4.11}
\]
Analogously, we can rewrite (4.8c) as
\[ \frac{x_n - p_n}{\tau} - L^*y_{n+1} \in Ap_n. \] (4.12)

The monotonicity of \( A \) together with the inclusion \(-L^*\bar{y} \in A\bar{x}\) from (4.2) now yields
\[
0 \leq \left( \frac{x_n - p_n}{\tau} - L^*y_{n+1} + L^*\bar{y}, p_n - \bar{x} \right) + \left( y_{n+1}, Lp_n - L\bar{x} \right).
\] (4.13)

Adding (4.11) and (4.13) yields
\[
0 \leq \frac{1}{\sigma} \langle y_{n+1} - q_{n+1}, q_{n+1} - \bar{y} \rangle + \langle Lp_n - L\bar{x}, q_{n+1} - y_{n+1} \rangle + \frac{a_{21}}{c_{21}} \langle Lp_n - L\bar{x}, q_{n+1} - \bar{y} \rangle + \frac{1}{\tau} \langle x_n - p_n, p_n - \bar{x} \rangle,
\] (4.14)

which combined with (4.9) gives
\[
Q_1(x_{n+1} - \bar{x}, y_{n+2} - \bar{y}) - Q_1(x_n - \bar{x}, y_{n+1} - \bar{y})
\leq \frac{a_{21}}{2\tau} \|p_n - x_n\|^2 + \frac{1}{\tau} \langle x_n - \bar{x}, p_n - x_n \rangle + \frac{c_{21}}{2\sigma} \|q_{n+1} - y_{n+1}\|^2
+ \frac{1}{\sigma} \langle y_{n+1} - \bar{y}, q_{n+1} - y_{n+1} \rangle + \frac{a_{21}}{c_{21}} \langle \bar{y} - y_{n+1}, Lp_n - L\bar{x} \rangle
+ \langle q_{n+1} - y_{n+1}, L\bar{x} - Lx_n \rangle + a_{21} \langle y_{n+1} - q_{n+1}, Lp_n - Lx_n \rangle
+ \frac{1}{\sigma} \langle y_{n+1} - q_{n+1}, q_{n+1} - \bar{y} \rangle + \langle Lp_n - L\bar{x}, q_{n+1} - y_{n+1} \rangle
+ \frac{a_{21}}{c_{21}} \langle Lp_n - Lx_n, q_{n+1} - \bar{y} \rangle + \frac{1}{\tau} \langle x_n - p_n, p_n - \bar{x} \rangle
= \left( \frac{a_{21}}{2\tau} + \frac{1}{\tau} \right) \|p_n - x_n\|^2 + \left( \frac{c_{21}}{2\sigma} + \frac{1}{\sigma} \right) \|q_{n+1} - y_{n+1}\|^2
+ \left( \frac{a_{21}}{c_{21}} + 1 - a_{21} \right) \langle q_{n+1} - y_{n+1}, Lp_n - Lx_n \rangle
= -Q_2(p_n - x_n, q_{n+1} - y_{n+1}).
\] (4.15)

This concludes the proof. \( \square \)

**Proposition 4.8.** Let \( Q_1 \) and \( Q_2 \) be defined as in Lemma 4.6 and assume the conditions stated in Theorem 4.3 hold.

1. If \( A \) is \( \mu_1 \)-strongly monotone for some \( \mu_1 > 0 \), then
\[
Q_1(x_{n+1} - \bar{x}, y_{n+2} - \bar{y}) - Q_1(x_n - \bar{x}, y_{n+1} - \bar{y}) + \mu_1 \|p_n - \bar{x}\|^2
\leq -Q_2(p_n - x_n, q_{n+1} - y_{n+1}).
\]

2. If \( B^{-1} \) is \( \mu_2 \)-strongly monotone for some \( \mu_2 > 0 \), then
\[
Q_1(x_{n+1} - \bar{x}, y_{n+2} - \bar{y}) - Q_1(x_n - \bar{x}, y_{n+1} - \bar{y}) + \mu_2 \|q_{n+1} - \bar{y}\|^2
\leq -Q_2(p_n - x_n, q_{n+1} - y_{n+1}).
\]
Proof. If $A$ is $\mu_1$-strongly monotone, we obtain from (4.12) and $-L^* \bar{y} \in A \bar{x}$ the estimation

$$\mu_1 \|\bar{x} - p_n\|^2 \leq \left\langle \frac{x_n - p_n}{\tau} - L^* y_{n+1} + L^* \bar{y}, p_n - \bar{x} \right\rangle,$$

which is a sharpened version of (4.13). By modifying (4.14) and (4.15) accordingly, we get the assumption. The case of a strongly monotone $B^{-1}$ is analogously shown by improving (4.11).

Having stated and proved the necessary estimations, we are now ready to prove Theorem 4.2.

Proof of Theorem 4.2 Let $(\bar{x}, \bar{y})$ satisfy (4.2). By Proposition 4.7, we get the estimation

$$Q_1(x_{n+1} - \bar{x}, y_{n+2} - \bar{y}) - Q_1(x_n - \bar{x}, y_{n+1} - \bar{y}) \leq -Q_2(p_n - x_n, q_{n+1} - y_{n+1}).$$

Considering Lemma 4.6, we see that the real sequence

$$(Q_1(x_n - \bar{x}, y_{n+1} - \bar{y}))_{n \geq 1}$$

is monotonically nonincreasing and therefore has a limit for each primal-dual solution $(\bar{x}, \bar{y})$. Furthermore, for all $N \geq 1$,

$$Q_1(x_N - \bar{x}, y_{N+1} - \bar{y}) - Q_1(x_0 - \bar{x}, y_1 - \bar{y}) \leq - \sum_{n=0}^{N-1} Q_2(p_n - x_n, q_{n+1} - y_{n+1}).$$

By Lemma 4.6 we have $Q_1(x_N - \bar{x}, y_{N+1} - \bar{y}) \geq 0$ and

$$Q_1(x_0 - \bar{x}, y_1 - \bar{y}) \geq \sum_{n=0}^{N-1} Q_2(p_n - x_n, q_{n+1} - y_{n+1})$$

$$\geq \sum_{n=0}^{N-1} C_2 \|p_n - x_n\|^2$$

as well as

$$Q_1(x_0 - \bar{x}, y_1 - \bar{y}) \geq \sum_{n=0}^{N-1} D_2 \|q_{n+1} - y_{n+1}\|^2.$$

Since this holds for arbitrary $N \geq 1$, this proves parts (a) and (b) of the theorem.

On the other hand, we have

$$Q_1(x_0 - \bar{x}, y_1 - \bar{y}) \geq Q_1(x_N - \bar{x}, y_{N+1} - \bar{y}) \geq C_1 \|x_N - \bar{x}\|^2$$

and

$$Q_1(x_0 - \bar{x}, y_1 - \bar{y}) \geq Q_1(x_N - \bar{x}, y_{N+1} - \bar{y}) \geq D_1 \|y_{N+1} - \bar{y}\|^2.$$
for all $N \geq 1$, so the sequences $(x_n)_n$ and $(y_n)$ are bounded in $\mathcal{X}$ and $\mathcal{Y}$, respectively. Let $(n_k)_k$ be a subsequence with $x_{n_k} \to x_\infty \in \mathcal{X}$ and $y_{n_k+1} \to y_\infty \in \mathcal{Y}$. By (4.12) and (4.10), we obtain

$$\frac{x_{n_k} - p_{n_k}}{\tau} - L^* y_{n_k+1} \in A p_{n_k},$$

$$\frac{y_{n_k+1} - q_{n_k+1}}{\sigma} + L p_{n_k} + \frac{a_{21}}{c_{21}} (L p_{n_k} - L x_{n_k}) \in B^{-1} q_{n_k+1}.$$ 

Now apply [6] Proposition 2.4] with

$$a_k = p_{n_k},$$

$$a_k^* = \frac{x_{n_k} - p_{n_k}}{\tau} - L^* y_{n_k+1},$$

$$b_k = \frac{y_{n_k+1} - q_{n_k+1}}{\sigma} + L p_{n_k} + \frac{a_{21}}{c_{21}} (L p_{n_k} - L x_{n_k}),$$

$$b_k^* = q_{n_k+1}$$

and observe that

$$a_k = x_{n_k} + (p_{n_k} - x_{n_k}) \to x_\infty,$$

$$b_k^* = y_{n_k+1} + (q_{n_k+1} - y_{n_k+1}) \to y_\infty,$$

$$a_k^* + L^* b_k^* = \frac{x_{n_k} - p_{n_k}}{\tau} + L^* (q_{n_k+1} - y_{n_k+1}) \to 0,$$

$$L a_k - b_k = - y_{n_k+1} + \frac{a_{21}}{c_{21}} (L p_{n_k} - L x_{n_k}) \to 0$$

because parts (4) and (5) imply that $x_{n_k} - p_{n_k} \to 0$ and $y_{n_k+1} - q_{n_k+1} \to 0$ as $k \to +\infty$. This gives $L x_\infty \in B^{-1} y_\infty$ and $-L^* y_\infty \in A x_\infty$, i.e., $(x_\infty, y_\infty)$ satisfies (4.2). Since the choice of the weakly convergent subsequence was arbitrary, each weak sequential cluster point satisfies (4.2). Claim (c) now follows from [9, Lemma 2.47] applied to the norm $\sqrt{Q_1()}$ on the product space $\mathcal{X} \times \mathcal{Y}$ and to the solution set of (4.2).

Now assume that $A$ is $\mu_1$-strongly monotone for some $\mu_1 > 0$. By Proposition 1.8 we get the estimation

$$Q_1(x_{n+1} - \bar{x}, y_{n+1} - \bar{y}) - Q_1(x_n - \bar{x}, y_{n+1} - \bar{y}) + \mu_1 \| p_n - \bar{x} \|^2 \leq -Q_2(p_n - x_n, q_{n+1} - y_{n+1})$$

for all $n \geq 0$. Choose $N \geq 1$ and sum up this inequality for $n = 0, \ldots, N - 1$ to obtain

$$Q_1(x_N - \bar{x}, y_{N+1} - \bar{y}) - Q_1(x_0 - \bar{x}, y_1 - \bar{y}) + \mu_1 \sum_{n=0}^{N-1} \| p_n - \bar{x} \|^2 \leq - \sum_{n=0}^{N-1} Q_2(p_n - x_n, q_{n+1} - y_{n+1})$$

Since the terms $Q_1(x_N - \bar{x}, y_{N+1} - \bar{y})$ and $\sum_{n=0}^{N-1} Q_2(p_n - x_n, q_{n+1} - y_{n+1})$ are nonnegative by Lemma 4.4, we obtain from

$$\mu_1 \sum_{n=0}^{N-1} \| p_n - \bar{x} \|^2 \leq Q_1(x_0 - \bar{x}, y_1 - \bar{y})$$
Analogously, one gets

\[ \mu^2 \sum_{n=0}^{N-1} \|q_{n+1} - \bar{y}\|^2 \leq Q_1(x_0 - \bar{x}, y_1 - \bar{y}), \]

and since \( N \) is arbitrary, both sums

\[ \sum_{n=0}^{\infty} \|p_n - \bar{x}\|^2 \quad \text{and} \quad \sum_{n=0}^{\infty} \|q_{n+1} - \bar{y}\|^2 \]

are finite. The uniqueness of the points \( \bar{x} \) under the assumption of strong monotonicity of \( A \) holds by the fact that we have shown \( p_n \to \bar{x} \) for any solution \((\bar{x}, \bar{y})\) of (4.2). An analogous argument for \( \bar{y} \) concludes the proof of Claim (d).

**Remark 4.9.** We were not able to show the weak convergence of PDHG (3.7) for \( \theta \neq 1 \) with this proof method. Indeed, by a straightforward calculation it can be shown that from Fejér monotonicity with respect to any quadratic form of the sequence \((x_n, y_{n+1})_n\) the conditions (4.6) can be derived, which implies \( \theta = 1 \).

### 4.3 Application to convex optimization

In this section, we specialize the scheme (4.8) to the case where the monotone operators are subdifferentials of proper, convex and lower semicontinuous functions. In this case, it is possible to get estimations for the Lagrangian, which is defined in (3.4).

**Theorem 4.10.** Given the assumptions in Theorem 4.2, let \( F : X \to \mathbb{R} \) and \( G : Y \to \mathbb{R} \) be two proper, convex and lower semicontinuous functions, \( A := \partial F \) and \( B := \partial G \). Let \( x \in X \) and \( y \in Y \) be arbitrary. Then the sequence \((q_n, y_n, p_n, x_n)\) generated by (4.8) satisfies

\[ \min_{n=0, \ldots, N-1} \left( \mathcal{L}(p_n; y) - \mathcal{L}(x; q_{n+1}) \right) \leq \frac{1}{N} Q_1(x_0 - x, y_1 - y), \]

\[ \mathcal{L} \left( \frac{1}{N} \sum_{n=0}^{N-1} p_n; y \right) - \mathcal{L} \left( x; \frac{1}{N} \sum_{n=0}^{N-1} q_{n+1} \right) \leq \frac{1}{N} Q_1(x_0 - x, y_1 - y) \]

This theorem is proved using the following proposition, which bounds the Lagrangian in terms of the quadratic forms defined in Lemma 4.6.

**Proposition 4.11.** Let \( F : X \to \mathbb{R} \) and \( G : Y \to \mathbb{R} \) be two proper, convex and lower semicontinuous functions, \( A = \partial F \) and \( B = \partial G \). Then, the Lagrangian satisfies

\[ \mathcal{L}(p_n; y) - \mathcal{L}(x; q_{n+1}) \leq Q_1(x_n - x, y_{n+1} - y) - Q_1(x_{n+1} - x, y_{n+2} - y) \]

\[ - Q_2(p_n - x_n, q_{n+1} - y_{n+1}) \]

for all \( n \geq 1, x \in X \) and \( y \in Y \).

15
Proof. Since $B^{-1} = \partial G^*$ and $A = \partial F$, the inclusions (4.10) and (4.12) provide certain subgradients, which imply the inequalities
\[
G^*(y) \geq G^*(q_{n+1}) + \frac{1}{\sigma} \langle y_{n+1} - q_{n+1}, y - q_{n+1} \rangle + \langle Lp_n, y - q_{n+1} \rangle \\
+ \frac{a_1}{c_1} \langle Lp_n - Lx_n, y - q_{n+1} \rangle, \\
F(x) \geq F(p_n) + \frac{1}{\tau} \langle x_n - p_n, x - p_n \rangle - \langle L^* y_{n+1}, x - p_n \rangle.
\]

Therefore, we have
\[
\mathcal{L}(p_n; y) - \mathcal{L}(x; q_{n+1}) \\
= \langle Lp_n, y \rangle + F(p_n) - G^*(y) - (Lx, q_{n+1}) - F(x) + G^*(q_{n+1}) \\
\leq \frac{1}{\tau} \langle x_n - p_n, p_n - x \rangle + \langle Lp_n - Lx, q_{n+1} - y_{n+1} \rangle \\
+ \frac{1}{\sigma} \langle y_{n+1} - q_{n+1}, q_{n+1} - y \rangle + \frac{a_1}{c_1} \langle Lp_n - Lx_n, q_{n+1} - y \rangle.
\]

The right-hand side is now (except for the replacement of $\bar{x}$ and $\bar{y}$ by $x$ and $y$, respectively) equal to the one in (4.14), and one easily checks by an analogous calculation, that it equals the expression in assertion.

Proof of Theorem 4.10. By summing the inequality in Proposition 4.11 for $n = 0, \ldots, N - 1$ and dividing by $N$ for some $N \geq 1$, we get
\[
\frac{1}{N} \sum_{n=0}^{N-1} (\mathcal{L}(p_n; y) - \mathcal{L}(x; q_{n+1})) \leq \frac{1}{N} Q_1(x_0 - x, y_1 - y)
\]
for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, where we dropped nonpositive terms on the right-hand side.

We have two possibilities to further estimate the left-hand side: First, we notice that it is the arithmetic mean of numbers, which is always greater than the minimum, i.e.,
\[
\frac{1}{N} \sum_{n=0}^{N-1} (\mathcal{L}(p_n; y) - \mathcal{L}(x; q_{n+1})) \geq \min_{n=0, \ldots, N-1} (\mathcal{L}(p_n; y) - \mathcal{L}(x; q_{n+1})).
\]

On the other hand, the Lagrangian is convex in its first and concave in its second component, so
\[
\frac{1}{N} \sum_{n=0}^{N-1} (\mathcal{L}(p_n; y) - \mathcal{L}(x; q_{n+1})) \geq \mathcal{L} \left( \frac{1}{N} \sum_{n=0}^{N-1} p_n; y \right) - \mathcal{L} \left( x; \frac{1}{N} \sum_{n=0}^{N-1} q_{n+1} \right). \quad \square
\]

4.4 Formulating the scheme using matrices

To conclude this section, we note that by introducing the intermediate states $w_n = c_{11} q_n + c_{12} y_n$ and $v_{n+1} = a_{11} p_n + a_{12} x_n$, and the $2 \times 2$ matrices $A, B, C, D$, the scheme presented in (1.1) can be written as
\[
\begin{bmatrix}
w_n \\
y_{n+1}
\end{bmatrix}
= (C \otimes \text{Id}) \ \text{diag}(\text{Prox}_{G^*}, \text{Id}) \ (B \otimes \text{Id}) \begin{bmatrix}
L v_n \\
y_n
\end{bmatrix} \\
\begin{bmatrix}
v_{n+1} \\
x_{n+1}
\end{bmatrix}
= (A \otimes \text{Id}) \ \text{diag}(\text{Prox}_{F^*}, \text{Id}) \ (D \otimes \text{Id}) \begin{bmatrix}
L^* w_n \\
x_n
\end{bmatrix}.
\]

For example the PDHG algorithm (3.7) is obtained by setting
\[ A = \begin{bmatrix} 1 + \theta & -\theta \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \sigma & 1 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} -\tau & 1 \\ 0 & 1 \end{bmatrix}, \]
(4.17)
and the Douglas-Rachford algorithm (3.8) by taking
\[ A = \begin{bmatrix} 2 & -1 \\ \lambda_n & 1 - \lambda_n \end{bmatrix}, \quad B = \begin{bmatrix} \sigma & 1 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} \lambda_n & 1 - \lambda_n \\ \lambda_n & 1 - \lambda_n \end{bmatrix}, \quad D = \begin{bmatrix} -\tau & 1 \\ 0 & 1 \end{bmatrix}. \]
(4.18)

5 Learning an optimization solver

Most optimization problems are solved using iterative methods, akin to the ones presented in Sections 3 and 4. However, the number of iterations it takes in order for the algorithm to converge is in general hard to predict, which creates problems in time-critical applications. In these situations one could instead consider only doing a predefined fixed number \( n \) of iterations. A natural question that arises in response to this is: what parameter values in the optimization solver give the best improvement of the objective function in \( n \) iterations? This question leads to a meta-optimization over optimization solvers. Moreover, in general we are not only interested in optimizing one single cost function, but rather a (potentially infinite) family \( \{F_\theta\}_\theta \) of cost functions. Hence, to make the question precise one needs to specify which family of optimization solvers one is considering, which is the family of cost functions of interested, and what is meant with “best improvement”.

One such question was raised in [21], where the authors consider the worst-case performance \( \sup_\theta [F_\theta(x_n) - F_\theta(\bar{x}_\theta)] \) of gradient-based algorithms over the set of continuously differentiable functions with Lipschitz-continuous gradients, and with a uniform upper bound on the Lipschitz constants. Subsequent work along the same lines can be found in [28, 43].

The idea of optimizing over optimization solvers has also been considered from a machine learning perspective. This has for example been done using reinforcement learning [32], and using unsupervised learning [7, 25]. In the latter category, one looks for algorithm parameters which minimize the expected value of the difference in objective function value,
\[
E_\theta [F_\theta(x_n) - F_\theta(\bar{x}_\theta)] = E_\theta [F_\theta(x_n)] - E_\theta [F_\theta(\bar{x}_\theta)]
\]
(5.1)
where \( \bar{x}_\theta \) is a minimum of \( F_\theta \) and \( x_n \) is the output of the algorithm after \( n \) iterations. However, optimizing (5.1) with respect to the parameters of the method is independent of the optimal points \( \{\bar{x}_\theta\}_\theta \), thus, this translates into unsupervised learning, i.e., the cost function \( E_\theta [F_\theta(x_n)] \) does not depend on \( \bar{x}_\theta \). In this setting, [7] restricts attention to an architecture that operates individually on each coordinate of \( x \). This is done in order to limit the number of parameters in the algorithm, which otherwise would grow exponentially with the dimension of \( x \). To overcome this, we use an approach similar to [25], where the network architecture is inspired by modern first-order optimization solvers for nonsmooth problems, as presented in Sections 3 and 4. Similar ideas have also recently been explored for supervised learning in inverse problems in [4, 5].
5.1 Unrolled gradient descent as a neural network

Before we define the architecture considered in this work, we first present an illustrative example. To this end, consider the optimization problem

$$\min_x F(x).$$

We assume that $F$ is smooth, which means that the problem can be solved using a standard gradient descent algorithm, i.e., by performing the updates

$$x_k = x_{k-1} - \alpha_k \nabla F(x_k).$$

The gradient descent algorithm contains a set of parameters that need to be selected, namely the step length for each iteration, $\alpha_k$. This is normally done via the Goldstein rule or backtracking line search (Armijo rule) \[12\], which under suitable conditions ensures convergence to the optimal point $\bar{x}$.

However, if we only run the algorithm for a fixed number $n$ of steps, the gradient descent algorithm can be seen as a recurrent neural network (RNN), as shown in Figure 1. Each level in the network performs the computation $x_{k-1} - \alpha_k \nabla F(x_{k-1})$ and the parameters of the network are $\alpha = [\alpha_1, \ldots, \alpha_n]$. For a given family $\{F_\theta\}_\theta$ of cost functions these parameters can be trained (optimized) by minimizing $E_\theta [F_\theta(x_n)]$, where $x_n$ is the output of the RNN in Figure 1. For simple cases this can be done analytically.

**Example 5.1.** Consider the family $(F_b)_b$ of functions $F_b : \mathbb{R}^n \to \mathbb{R}$ given by

$$F_b(x) = \frac{1}{2} x^T Ax - b^T x,$$

where $A \in \mathbb{R}^{n \times n}$ is a (fixed) symmetric and positive definite matrix. The global minimum of $F_b$ is given by $\bar{x} = A^{-1}b$. Denote by $\Lambda_\sigma$ the result of taking a gradient step of length $\sigma > 0$, i.e.,

$$\Lambda_\sigma(x) = x - \sigma \nabla F_b(x) = x - \sigma (Ax - b), \quad x \in \mathbb{R}^n.$$

Let $x_0 \in \mathbb{R}^n$ be an arbitrary starting point of the iteration. We can expand

$$F_b(\Lambda_\sigma(x_0)) = F_b(x_0 - \sigma (Ax_0 - b))$$

$$= \frac{1}{2} (x_0 - \sigma (Ax_0 - b))^T A(x_0 - \sigma (Ax_0 - b)) - b^T (x_0 - \sigma (Ax_0 - b))$$

$$= \frac{\sigma^2}{2} (Ax_0 - b)^T A (Ax_0 - b) - \sigma \| Ax_0 - b \|^2 + F_b(x_0).$$
Let \( b \) be a random variable distributed according to \( b \sim \mathcal{P} \) for some probability distribution \( \mathcal{P} \) with finite first and second moments. Finding a \( \sigma \) that minimizes the expectation

\[
E_{b \sim \mathcal{P}} \left[ F_b(A_x(x_0)) \right] = \frac{\sigma^2}{2} E_{b \sim \mathcal{P}} \left[ (Ax_0 - b)^\top A(x_0 - b) \right] - \sigma E_{b \sim \mathcal{P}} \left[ \|Ax_0 - b\|^2 \right] + E_{b \sim \mathcal{P}} \left[ F_b(x_0) \right],
\]

is a quadratic problem in one variable, and the optimal value of \( \sigma \) is thus

\[
\sigma = \frac{E_{b \sim \mathcal{P}} \left[ \|Ax_0 - b\|^2 \right]}{E_{b \sim \mathcal{P}} \left[ (Ax_0 - b)^\top A(x_0 - b) \right] - \sigma E_{b \sim \mathcal{P}} \left[ \|b\|^2 \right].
\]

In some particular cases this expression can be simplified. For example if \( A = I \), then \( \sigma = 1 \) as expected. Or if \( x_0 = 0 \), then \( \sigma = E_{b \sim \mathcal{P}}[\|b\|^2] / E_{b \sim \mathcal{P}}[b^\top Ab] \).

### 5.2 Parameterizing a family of optimization algorithms

Similarly to the considerations in Section 5.1, one can consider (4.16) for a fixed number of iterations as an RNN. In this case, the parameters of the network are the elements of the matrices \( A, B, C, \) and \( D \), and this structure can easily be extended in order to incorporate more memory in the network. In this work we assume that the computationally expensive part of the algorithm is the evaluation of the operator \( L \) and its adjoint, which is typically the case in inverse problems in imaging, e.g., three-dimensional CT [36, 37]. Therefore, the extension presented here thus keeps one evaluation \( L \) and one evaluation of \( L^* \) in each iteration.

To this end, let \( N \) be the number of primal variables \( x^1, \ldots, x^N \in \mathcal{X} \) and \( M \) be the number of dual variables \( y^1, \ldots, y^M \in \mathcal{Y} \). Moreover, by \( \otimes \) we denote the Kronecker product, and by \( \text{diag}(A, B, \ldots) \) we denote the diagonal operator with \( A, B, \ldots \) on the diagonal. The iterations in (4.16) are extended to yield (5.2), which is parameterized by the four sequences of matrices \( A_n, D_n \in \mathbb{R}^{N \times N} \) and \( B_n, C_n \in \mathbb{R}^{M \times M} \).

\[
\begin{bmatrix}
y_n^{1+1} \\
y_n^{2+1} \\
\vdots \\
y_n^{M+1}
\end{bmatrix}
= (C_n \otimes \text{Id}) \ \text{diag}(\text{Prox}_{\sigma_n^2, \text{Id}^M}, \text{Id}^N) \ \begin{bmatrix}
L x^1_n \\
y_n^{2} \\
\vdots \\
y_n^{M}
\end{bmatrix},
\]

\[
\begin{bmatrix}
x_n^{1+1} \\
x_n^{2+1} \\
\vdots \\
x_n^{N+1}
\end{bmatrix}
= (A_n \otimes \text{Id}) \ \text{diag}(\text{Prox}_{\tau_n^2, \text{Id}^N}, \text{Id}^M) \ \begin{bmatrix}
L^* y_n^{1+1} \\
x_n^{2} \\
\vdots \\
x_n^{N}
\end{bmatrix}. \tag{5.2}
\]
5.2.1 Extension to forward-backward-forward methods

Some methods in the literature, so called forward-backward-forward methods, include an extra evaluation of the operator and its adjoint per iteration, see, e.g., [14] [19]. However, since this evaluation is assumed to be the expensive part in our setting we consider this as two iterations. Thus, if we start with the x-iterate and allow for two iterations in our framework to complete one iteration in such a framework, our proposed algorithm contains, e.g., [19, Equation (4.5)]. Letting denote an element that can take any value, one such set of matrices are given by

\[
D_{2n-1} = \begin{bmatrix}
-\gamma_n & 0 & 0 & 1 \\
-\gamma_n & 0 & 0 & 1 \\
\ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad A_{2n-1} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
\ldots & \ldots & \ldots & \ldots \\
1 & 0 & 0 & 0
\end{bmatrix},
\]

\[
B_{2n-1} = \begin{bmatrix}
\gamma_n & 0 & 0 & 1 \\
\gamma_n & 0 & 0 & 1 \\
\ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad C_{2n-1} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0
\end{bmatrix},
\]

for the odd iterations and

\[
D_{2n} = \begin{bmatrix}
\ldots & \ldots & \ldots & \ldots \\
0 & -1 & 1 & 0 \\
-\gamma_n & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad A_{2n} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
\ldots & \ldots & \ldots & \ldots \\
0 & 1 & 1 & 0
\end{bmatrix},
\]

\[
B_{2n} = \begin{bmatrix}
\ldots & \ldots & \ldots & \ldots \\
0 & -1 & 1 & 0 \\
\gamma_n & 0 & 0 & 1 \\
\ldots & \ldots & \ldots & \ldots
\end{bmatrix}, \quad C_{2n} = \begin{bmatrix}
0 & 1 & 1 & 0 \\
\ldots & \ldots & \ldots & \ldots \\
0 & 1 & 1 & 0
\end{bmatrix},
\]

for the even iterations.

Remark 5.2. Other forward-backward-forward methods have been proposed in the literature, some of which are general enough to include the PDHG as a special case [26], or both the PDHG and the Douglas-Rachford algorithm as special cases [31]. However, these methods include a step-length computation in their updates. This computation involves evaluating the norm of current iterates, which is not possible to achieve by only doing the linear operations we propose. Of course, allowing the matrix elements to be nonlinear functions of the states would allow us to incorporate also these methods, however, that is beyond the scope of this article.

6 Numerical experiments

We demonstrate the above presented scheme by considering a numerical example in computerized tomography. In this example we let T be the Radon transform and consider TV regularization. This means that the family of objective functions is given by

\[
H_b(x) = \|T(x) - b\|_2^2 + \lambda \|\nabla x\|_1,
\]

(6.1)
i.e., the family is parameterized by the data $b$. Throughout all examples, the size of the image $x$ is $512 \times 512$ pixels, and the regularization parameter $\lambda > 0$ is fixed. We train and evaluate several of the algorithms described in this article on a clinically realistic data set, namely simulated data from human abdomen CT scans as provided by Mayo Clinic for the AAPM Low Dose CT Grand Challenge [34]. Examples of two-dimensional phantoms from this data set are given in Figure 2. The data is generated by applying $T$ to the phantoms and then adding 5% white Gaussian noise.

![Figure 2: Several examples of phantoms used for generating data. These phantoms take values between $[0, 3.13]$, but all images are shown using a window set to $[0, 8, 1.2]$ in order to enhance contrast of clinically more relevant details.](image)

### 6.1 Implementation and specifications of the training

For each algorithm, the number of unrolled iterations, corresponding to the depth of the network, was set to $n_{\text{max}} = 10$, and all evaluations have been done with this depth. However, in order to heuristically induce better stability of the general schemes, we have trained using a stochastic depth as follows: In each step of the training the depth of the network has been set to the outcome of the stochastic variable

$$n_{\text{max}} = \min \left[ \text{round}(8 + Z), 100 \right].$$

In this expression, $Z$ is a log-normal stochastic variable. Moreover, since this is a heavy-tailed distribution, and due to limits in computational resources, the min was used to limit the depth of the network to 100. The underlying normal distribution has been selected to have standard deviation $1.25^2/2$, which ensures that $\mathbb{E}[n_{\text{max}}] \approx \mathbb{E}[8 + Z] = 10$.

In order to improve stability and generalization properties of the trained networks we have normalized the operators before training, i.e., rescaled them so that $\|T\|_2 = \|\nabla\|_2 = 1$. For the same reasons, we have used the zero vector as initial guess for all networks. Training has been done using the ADAM solver [29], with standard parameter values except for $\beta_2 = 0.99$. Moreover, we have used gradient clipping to limit the norm of the gradient of the training cost function (5.1) to be less than or equal to one [40]. As step length (learning rate) we have used a cosine annealing scheme [33], i.e., a step length which in step $t$ takes the value

$$\eta_t = \eta_0 \left( 1 + \cos \left( \pi \frac{t}{t_{\text{max}}} \right) \right)^{\frac{1}{2}},$$

21
where the initial step length $\eta_0$ has been set to $10^{-3}$. We have trained for $t_{\text{max}} = 100\,000$ steps and have used 9 out of 10 phantoms from the AAPM Low Dose CT Grand Challenge for training and one for evaluation.

All algorithms have been implemented using ODL [3], the GPU accelerated version of ASTRA [1, 39], and Tensorflow [2]. The source code to replicate the experiments is available online. We have used this setup to train the following methods.

**PDHG method.** This corresponds to optimal selection of the parameters $\theta$, $\tau$, and $\sigma$ for the PDHG method (3.7) on the family of cost functions (6.1). In order to achieve this we need to enforce the constraints $\theta \in [0, 1]$ and $\sigma \tau \|L\|^2 < 1$. This has been done implicitly by a change of variables, namely by

$$
\theta = \frac{e^{s_1}}{1 + e^{s_1}}, \quad \tau = \frac{1}{\|L\|} \cdot \frac{e^{s_2 + s_3}}{1 + e^{s_2}}, \quad \sigma = \frac{1}{\|L\|} \cdot \frac{e^{s_2 - s_3}}{1 + e^{s_2}}
$$

with $s_1, s_2, s_3 \in \mathbb{R}$. Here, $s_2$ determines how close the parameters $\sigma$ and $\tau$ are to the constraint $\sigma \tau \|L\|^2 < 1$, while $s_3$ determines the trade-off between $\tau$ and $\sigma$.

**PDHG method without constraints on the parameters.** Here we train the same parameters $\theta, \tau, \sigma$ as in the PDHG method. However, we do not make the change of variables (6.2), therefore, no constraints on $\theta$, $\tau$, and $\sigma$ are enforced in the training. This means that the resulting scheme might not correspond to a globally convergent optimization algorithm.

**Proposed method from Section 4.** This corresponds to optimal parameter selection for the method (4.8) on the family of cost functions (6.1). To adhere to the constraints in the assumptions in Theorem 4.2, we have used the same kind of variable change as in (6.2), namely

$$
a_{21} = \frac{2e^{s_1}}{1 + e^{s_1}}, \quad c_{21} = \frac{2e^{s_2}}{1 + e^{s_2}}, \quad \tau = \frac{K}{\|L\|} \cdot \frac{e^{s_3 + s_4}}{1 + e^{s_3}}, \quad \sigma = \frac{K}{\|L\|} \cdot \frac{e^{s_3 - s_4}}{1 + e^{s_3}}
$$

with $s_1, \ldots, s_4 \in \mathbb{R}$, where $K = \frac{c_{21}^2(2-a_{21})(2-c_{21})}{(a_{21} + c_{21} - a_{21} c_{21})^2}$, as in (4.7).

**Parameterization proposed in Section 5.2.** Here, we have trained schemes of the form (5.2). We have done this for constant sequences of matrices $A$, $B$, $C$, and $D$. We restricted ourselves to the sizes $N = M = 2$ and $N = M = 3$.

### 6.2 Performance of the trained methods

Numerical results are presented in Figure 3 which shows the objective function value (6.1) as a function of the iteration number. To obtain an estimation of the true optimal value of (6.1), we have run 1 000 iterations of PDHG with parameters as in [42]. In Figure 3 we show the difference between the obtained loss and the minimal loss, averaged over 100 samples. As can be seen, the scheme proposed in Section 5 with $N = M = 3$ performs best at 10 iterations. Moreover,
a general trend seems to be that more parameters in the algorithms improve
the performance. Finally, the results from one specific phantom are presented
as reconstructions in Figure 4. Note that the reconstruction by PDHG with
parameters as in [42] is left out due to the page layout.

Figure 3: The figure shows the objective function values of several reconstruction
methods as a function of the iteration number. Solid lines are real optimization
solvers, dotted lines are schemes that might not converge to the true optimal
solution. (i) PDHG with parameters as in Sidky et al. [42], (ii) PDHG with
trained parameters with constraints, (iii) proposed solver (4.8) with trained
parameters, (iv) PDHG with trained free parameters, (v) proposed scheme (5.2)
with $N = M = 2$, and (vi) proposed scheme (5.2) with $N = M = 3$.

6.2.1 Generalization to other iteration numbers

Figure 3 also shows how well the learned algorithms generalize with iteration
number. For the trained, convergent solvers, the objective function value keeps
decreasing as expected. Furthermore, the solver proposed in (4.8) performs
better than the others also when the number of iterations are increased, but
poorer in the beginning. For the other schemes, it can be noted that, while
training more parameters seems to increase the performance after 10 iterations,
it also seems to decrease the generalizability of the algorithm with respect to
an increase in the number of iterations.
6.2.2 Generalization to deblurring

Next, we investigate the generalizability of the trained networks to other optimization problems by replacing the forward operator $T$ in (6.1) with a convolution. This corresponds to another TV problem in imaging, namely image deblurring.

Clearly, the trained networks that correspond to optimization solvers with convergence guarantees can be applied to other convex optimization problems. (Note that we still normalize the operators to have operator norm one so that the assumptions in Theorem 4.2 do not change.) However, nothing guarantees that parameters that give fast convergence on one type of problems will also give fast convergence on another one.

Two example images are shown in Figure 5. The images corresponding to the “Raccoon” test image, are of size $1024 \times 768$ and use a different regularization parameter. Blurring has been done with a Gaussian kernel, and then 5% white noise has been added to the images. Again, to obtain an estimation of the true optimal value of we have run 1000 iterations of PDHG with parameters as in [42]. For each algorithm, the difference between the obtained
loss and minimal loss is presented in Table 1 and the deblurred images are shown in Figures 6 and 7. Again, the reconstruction by PDHG with parameters as in [42] is left out due to the page layout.

The $3 \times 3$ method does not generalize well. However, the $2 \times 2$ method generalizes, and the optimization algorithm from Section 4 with trained parameters, is one of the best on these two test problems.

![Example images used for the deblurring problem in Section 6.2.2.](image)

Table 1: Loss function values for the deblurring problem in Section 6.2.2.

<table>
<thead>
<tr>
<th>Method</th>
<th>$H_b(x_{10})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDHG with parameters from [42]</td>
<td>Ascent</td>
</tr>
<tr>
<td>Trained PDHG with constraints on parameters</td>
<td>5.514</td>
</tr>
<tr>
<td>Trained solver [4.5]</td>
<td>4.256</td>
</tr>
<tr>
<td>Trained PDHG without constraints on parameters</td>
<td>2.204</td>
</tr>
<tr>
<td>Trained scheme of type (5.2) with $N = M = 2$</td>
<td>2.173</td>
</tr>
<tr>
<td>Trained scheme of type (5.2) with $N = M = 3$</td>
<td>208.37</td>
</tr>
</tbody>
</table>

Acknowledgments

We acknowledge Swedish Foundation of Strategic Research grants AM13-0049 and ID14-0055, Swedish Research Council grant 2014-5870 and support from
Figure 6: Reconstructions with the trained algorithms for the “Ascent” image.

Elekta.

The authors thank Dr. Cynthia McCollough, the Mayo Clinic, and the American Association of Physicists in Medicine for providing the data necessary for performing comparison using a human phantom.

References


Figure 7: Reconstructions with the trained algorithms for the “Raccoon” image.


