Optimizing First-Order Method Parameters via Differentiation of the Performance Estimation Problem

Anton Åkerman

Department of Automatic Control

MSc Thesis
TFRT-6204
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

This thesis treats the problem of finding optimal parameters for first-order optimization methods. In part, we use the Performance Estimation Problem (PEP), a framework for convergence analysis of first-order optimization methods. The fundamental idea of the PEP is to formulate the problem of finding the worst-case convergence rate of a first-order optimization algorithm, as an optimization problem. We also use recent methods for differentiating convex optimization problems. The goal is to explore the use of gradient-based methods for finding optimal parameters of first-order optimization methods, within the context of the Performance Estimation Problem. By differentiating the PEP, we can find gradients which can be used in an attempt to search for optimal method parameters.

We consider the state space representation of first-order methods, which include many well-known first-order operator splitting methods. We propose a gradient-based algorithm for optimizing first-order method parameters, based on the differentiation algorithm from [Agrawal et al., 2020] and the PEP representations from [Upadhyaya et al., 2023], and show decent results. This is a heuristic approach to a non-convex optimization problem, but it works well for the Douglas–Rachford and Davis–Yin operator splitting methods. The results seem to agree with the theoretically optimal parameters for the Douglas–Rachford method, and the obtained convergence rates for the Davis–Yin method are better than the ones found in [Upadhyaya et al., 2023], using fixed parameters. The presented results, concern only those two methods, but the proposed algorithm is general. Based on some limited testing, this problem seems sensitive to numerical inaccuracy, and as a consequence, our approach using more exact gradients seems to outperform the built-in solver from SCIPY, which uses approximate gradients, terminating faster with comparable (or better) accuracy.

Acknowledgements

This thesis would not be possible without my supervisor, Pontus Giselsson. I would also like to thank all the people who brightened my days during my time writing this.

Contents

1

Introduction

First-order optimization methods are fundamental for large-scale optimization. Compared to second-order methods, the computational cost scales well with the problem size and they are relatively simple to implement. But for a method to be practical, it is also critical to understand the convergence properties, both in order to know for which problems it is suitable and for which set of method parameters it performs well. The goal of this project is to investigate gradient-based methods for optimizing first-order method parameters. For that purpose, we combine two recent developments in the field of convex optimization: The worst-case convergence for first-order methods is firstly formulated as a semidefinite program via the Performance Estimation Problem (PEP). Secondly, we find accurate gradients, using a method for differentiating optimization problems, proposed by [Agrawal et al., 2020].

The Performance Estimation Problem

The origin of the Performance Estimation Problem is [Drori and Teboulle, 2013], where the PEP framework for finding upper bounds on the worst-case convergence rate for a first-order method, is presented. The idea is to formulate the worst-case convergence as an optimization problem that can ultimately be reformulated into a semidefinite program. Through a series of following papers, the PEP framework has been developed further. An important contribution is an article on the interpolation of convex functions, [Taylor et al., 2016]. They give clear conditions for when the obtained convergence rate via the PEP is exact. Later, [Ryu et al., 2020] present a PEP framework for finding optimal parameters of first-order methods and use this to derive new tight contraction factors for the Douglas–Rachford operator splitting method. The PEP formulation used in our thesis is based on the state space formulation of first-order methods, a formulation that was first presented in [Lessard et al., 2016] and then extended in [Upadhyaya et al., 2023], where a generalized approach for finding linear and sublinear convergence rates for general quadratic Lyapunov functions, is introduced. The power of the performance estimation problem lies in the fact that it gives tight bounds on convergence rates, even in cases where the best-known analytical bounds are far from exact, all at the cost of solving relatively small semidefinite programs.

Differentiating convex optimization problems

The PEP approach will be combined with methods for differentiating conic programs. The basis is a generalization of the KKT conditions called the self-dual embedding, which can serve both as an optimality guarantee for a proposed solution and as proof of infeasibility. The use of differentiation to iteratively improve approximate solutions of the self-dual embedding was first proposed in [Busseti et al., 2018], with the purpose of improving solvers for conic optimization. As a follow-up, [Agrawal et al., 2020] showed that this could be extended to the differentiation of conic optimization problems if combined with the implicit function theorem.

Goal

The Performance Estimation Problem defines some function, $\mathcal{P}(\theta)$, describing the worst-case convergence rate for some optimization method, which depends on the method parameters, θ. This function is defined implicitly, via an optimization problem. We will make an attempt at finding the optimal parameters and convergence rates

$$\min_{\theta} \mathcal{P}(\theta). \tag{1.1}$$

For this purpose, efficient solvers often rely on gradients. Therefore, we differentiate the PEP, using an altered version of the algorithm developed in[Agrawal et al., 2020], in order to find gradients of the function $\mathcal{P}(\theta)$. This problem of selecting optimal parameters has been studied also in [Ryu et al., 2020], where a derivative-free solver is used, and the problem is separated as

$$\min_{\lambda,\gamma} \mathcal{P}(\lambda,\gamma) = \min_{\gamma} \left(\min_{\lambda} \mathcal{P}(\lambda,\gamma) \right). \tag{1.2}$$

This works well for methods with two parameters, but the time complexity is exponential in the number of parameters. Hopefully, gradient-based methods can give decent results also for higher dimensional problems, where this derivative-free method would not work. In this thesis, some three-parameter methods are tested as well, with good results, but problems with higher dimensions than so were not included in the report, due to limited time. This is a heuristic approach to the problem of optimizing method parameters, but the results seem decent, given sensible initial parameter choices and simple first-order methods.

Outline

The structure of the report will be as follows: Firstly, in chapter 3, the state space formulation of first-order methods will be summarized and the methods to be investigated are introduced. We will furthermore describe the conditions for the fixed point encoding property, i.e the guarantee that fixed points of an algorithm correspond to solutions for any given convex problem and vice versa. In the last section of the chapter, we give a small result on the possible structure of first-order methods with the fixed-point encoding property.

The formulation of the PEP we will use is described in chapter 4, where we describe a simplified PEP outline and then move on to the formulation of the PEP problem to be used in the report.

The differentiation of convex optimization problems is detailed in chapter 5, where we will summarize the methods introduced by [Agrawal et al., 2020].

In chapter 6, the previous theory is combined. We show how to differentiate the PEP problem specified in the previous chapter, for general state space representable methods. This is followed by some details concerning the implementation.

Our focus in this report is twofold. Firstly we investigate the use of gradient-based methods for finding parameters that minimize the convergence rate obtained through the PEP. For this purpose, the methods from [Agrawal et al., 2020] are used, with some adaptations necessary for this specific problem.

Secondly, we use this to find seemingly optimal parameters for the Douglas–Rachford and Davis–Yin methods, both in cases where the optimal parameters are known analytically and in some cases where they are not. The numerical results are summarized in section 7. A natural comparison is the use of built-in optimization methods, which often rely on a second-order, quasi-Newton method, like BFGS, as mentioned in [Ryu et al., 2020], with gradients obtained through a finite difference approximation. The main first-order methods we investigate are the Douglas-Rachford operator splitting method and the Davis-Yin three-operator splitting method. The method we propose seems computationally efficient and relatively accurate, for the relatively simple examples which are tested.

2

Prerequisites

We start by introducing some important notation, definitions and theorems used in later chapters. For further reading, we refer to [Boyd and Vandenberghe, 2011] and [Parikh, 2014].

Throughout we will consider convex functions, $f : \mathcal{H} \to \overline{\mathbb{R}}$, from a Hilbert space $\mathcal{H}$ to the set of extended real numbers, $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. For all purposes in this thesis, we will investigate functions $f : \mathbb{R}^n \to \overline{\mathbb{R}}$.

DEFINITION 2.1
The domain of $f : \mathcal{H} \to \overline{\mathbb{R}}$ is defined as

$$\operatorname{dom} f = \{x \mid f(x) < \infty\}. \tag{2.1}$$

DEFINITION 2.2
The function $f : \mathcal{H} \to \overline{\mathbb{R}}$ is said to be convex if

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \quad \forall x, y \in \mathcal{H} \text{ and } \lambda \in [0,1], \tag{2.2}$$

with the interpretation that the inequality always holds if the right-hand side is infinite.

DEFINITION 2.3
A convex function, $f : \mathcal{H} \to \overline{\mathbb{R}}$, for which the domain is non-empty, is called proper, and if the set

$$\{(x,r) \in \mathcal{H} \times \mathbb{R} \mid f(x) \leq r\} \tag{2.3}$$

is closed for all $r \in \mathbb{R}$, f is said to be closed convex.

DEFINITION 2.4
The subdifferential of a function, $f : \mathcal{H} \to \overline{\mathbb{R}}$, at a point $x \in \mathcal{H}$ is defined as

$$\partial f(x) = \{s \in \mathcal{H} \mid f(y) \geq f(x) + \langle s, y - x \rangle \quad \forall y \in \mathcal{H}\}. \tag{2.4}$$

If f is differentiable at $x \in \mathcal{H}$, then $\partial f(x) = \{\nabla f(x)\}$.

REMARK 2.5 For a proper, closed convex function $f : \mathcal{H} \to \overline{\mathbb{R}}$, Fermat's rule says that $x \in \mathcal{H}$ minimizes f if and only if $0 \in \partial f(x)$. This follows from the definition since

$$0 \in \partial f(x) \iff f(y) \geq f(x) + 0 \qquad \forall y \in \mathcal{H}. \tag{2.5}$$

The concepts of strongly convex and smooth functions will be used extensively throughout this thesis

DEFINITION 2.6
A convex function, $f : \operatorname{dom} f \to \mathbb{R}$, is said to be σ-strongly convex if for some $\sigma > 0$, the function $f(x) - \frac{\sigma}{2}||x||^2$ is convex.

DEFINITION 2.7
A function $f : \operatorname{dom} f \to \mathbb{R}$ is said to be β-smooth if it is differentiable and the gradient is Lipschitz continuous with Lipschitz constant β, i.e

$$||\nabla f(y) - \nabla f(x)||_2 \leq \beta ||y - x||_2 \tag{2.6}$$

Throughout, we will denote the set of σ-strongly convex, β-smooth functions $\mathcal{F}_{\sigma,\beta}$, where we say that $\sigma = 0$ or $\beta = \infty$ for functions which are not strongly convex nor smooth respectively. Oftentimes, the relevant quantity is the condition number, $\kappa = \frac{\beta}{\sigma}$, since the problem of finding x that minimizes $f \in \mathcal{F}_{\sigma,\beta}$ is equivalent to finding x which minimizes the normalized function: $\frac{f}{\sigma} \in \mathcal{F}_{1,\kappa}$.

DEFINITION 2.8
A set $S \subset \mathcal{H}$ is a cone if for each $x \in S$ also $\lambda x \in S$ for all $\lambda \geq 0$.

DEFINITION 2.9
The dual cone of a set $\mathcal{K}$ is denoted $\mathcal{K}^*$ and is defined as

$$\mathcal{K}^* = \{y \in \mathcal{H} \mid \langle x, y \rangle \geq 0, \, \forall x \in \mathcal{K}\}. \tag{2.7}$$

A cone, $\mathcal{K}$, which is equal to its dual cone, $\mathcal{K}^*$ is called self-dual.

DEFINITION 2.10
We define the proximal operator as

$$\operatorname{prox}_{\gamma f}(x) = \operatorname*{argmin}_{z} \left(f(z) + \frac{1}{2\gamma} ||z-x||_2^2, \right) \tag{2.8}$$

for each $x \in \mathcal{H}$ and where $\gamma > 0$.

For a proper, closed convex function, the proximal operator is always well-defined and unique. In that case it can alway be written in terms of the subdifferential

$$\begin{aligned} & 0 \in \partial \left(f(z) + \frac{1}{2\gamma} \|z-x\|^2 \right) \\ \iff & 0 \in \partial f(z) + \frac{1}{\gamma}(z-x) \\ \iff & \frac{x-z}{\gamma} \in \partial f(z). \end{aligned} \tag{2.9}$$

Proximal operators, along with gradients, are building blocks of the first-order methods we will consider.

THEOREM 2.11—MOREAU'S IDENTITY
Let $f : \mathcal{H} \to \overline{\mathbb{R}}$ be a proper closed convex function and let the dual function of f, f^*, be defined as

$$f^*(s) = \sup_x s^T x - f(x). \tag{2.10}$$

Then the Moreau identity states that

$$\operatorname{prox}_f(z) + \operatorname{prox}_{f^*}(z) = z \tag{2.11}$$

This holds in particular if f is the indicator function of a closed convex set $C \in \mathcal{H}$

$$f : \mathcal{H} \to \overline{\mathbb{R}} = \iota_C(x) = \begin{cases} 0 & \text{if} \quad x \in C \\ \infty & \text{otherwise} \end{cases} \tag{2.12}$$

$\iota_C^*(x) = \iota_{-C^*}$, where C^* is the dual cone of C.

PROPOSITION 2.12
The proximal operator of an indicator function for a closed convex set, $C \in \mathcal{H}$, is equivalent to orthogonal projection onto C, $\operatorname{prox}_{\iota_C}(z) = \Pi_C(z)$.

Proof. This follows from the definition of the proximal operator (2.8) since

$$\operatorname{prox}_{\gamma(\iota_C)}(z) = \operatorname*{argmin}_{x} \quad \iota_C(x) + \frac{1}{2\gamma}\|z-x\|^2 = \operatorname*{argmin}_{x \in C} \quad \frac{1}{2\gamma}\|z-x\|^2 = \Pi_C(z). \tag{2.13}$$

□

It follows as a consequence of the Moreau identity that

$$\Pi_C(z) + \Pi_{-C^*}(z) = z \iff \Pi_{-C^*}(z) = z - \Pi_C(z), \tag{2.14}$$

which will be used later. Moreover, it is worth noting that the orthogonal projection onto a closed convex set is Lipschitz continuous with Lipschitz constant 1.

The set of square real-valued matrices forms a vector space with the inner product

$$\langle A, B \rangle = \mathrm{Tr}(A^T B), \tag{2.15}$$

where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix. This is equivalent to the sum of the coordinate-wise multiplication

$$\mathrm{Tr}(A^T B) = \sum_{i,j} A_{ij} B_{ij} = \mathrm{vec}(A)^T \mathrm{vec}(B). \tag{2.16}$$

The notation $\mathrm{vec}(A)$ is used to represent reshaping the matrix $A \in \mathbb{R}^{n\times n}$ into a column vector $\mathrm{vec}(A) \in \mathbb{R}^{n^2}$.

PROPOSITION 2.13
The set of symmetric positive semidefinite (PSD) matrices, that we denote by $\mathbb{S}^n$, is a cone.

Proof. A symmetric matrix is PSD if and only if all eigenvalues are non-negative. If A has non-negative eigenvalues, so will λA, for each non-negative value of λ. □

PROPOSITION 2.14
The cone of PSD matrices is self-dual. A proof of this statement can be read in [Boyd and Vandenberghe, 2011].

DEFINITION 2.15
A real square matrix, $A \in \mathbb{R}^{n\times n}$ is said to be skew-symmetric if $A^T = -A$.

PROPOSITION 2.16
Let $A \in \mathbb{R}^{n\times n}$ be skew symmetric. Then the quadratic expression $u^T A u$ is zero, for any vector $u \in \mathbb{R}^n$.

Proof. Since A is skew-symmetric and $u^T A u$ is a scalar, we find that

$$u^T A u = \left(u^T A u\right)^T = -u^T A u. \tag{2.17}$$

□

This proves that $u^T A u$ is zero.

The Kroenecker product between $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, denoted by $\otimes$, is defined as

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{n1}B \\ a_{21}B & a_{22}B & \dots & a_{n1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix} \in \mathbb{R}^{mp\times nq}. \tag{2.18}$$

The Hadamard product for $A, B \in \mathbb{R}^{m\times n}$, $\circ$, is defined as

$$A \circ B = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \dots & a_{mn}b_{mn} \end{pmatrix} \tag{2.19}$$

We use $\mathrm{Ran}(A)$ to denote the range of $A \in \mathbb{R}^{m\times n}$,

$$\mathrm{Ran}(A) = \{y \in \mathbb{R}^m \mid y = Ax \text{ for some } x \in \mathbb{R}^n\}, \tag{2.20}$$

and $\mathrm{Null}(A)$ to denote the nullspace of $A \in \mathbb{R}^{m\times n}$,

$$\mathrm{Null}(A) = \{x \in \mathbb{R}^n \mid Ax = 0\}. \tag{2.21}$$

Lastly, we state the following theorems, which are fundamental to the differentiation described in chapter 5.

THEOREM 2.17—RADEMACHER

If f is a Lipschitz continuous function in $\mathbb{R}^n$, then f is differentiable almost everywhere. This holds in particular for projections onto closed convex sets.

Borrowing the notations used [Dontchev and Rockafellar, 2009], where the following version of the classical implicit function theorem is presented, we will let $D_x f$ denote the derivative operator for a function, f, with regard to x. In this case it should be interpreted as a Jacobian.

THEOREM 2.18—DINI'S IMPLICIT FUNCTION THEOREM

Let $f(p,x) : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable in a neighborhood of $(\bar{p},\bar{x}) \in \mathbb{R}^m \times \mathbb{R}^n$, for which $f(\bar{p},\bar{x}) = 0$. If the Jacobian of f w.r.t. x, $D_f^x(\bar{p},\bar{x})$, is non-singular then the solution mapping $S(p)$, defined as

$$S : p \in \mathbb{R}^m \mapsto \{x \in \mathbb{R}^n \mid f(p,x) = 0\}, \tag{2.22}$$

is single-valued and continuously differentiable in a neighborhood of $\bar{p}$. The Jacobian of S is

$$DS(p) = -D_x(f(p,S(p)))^{-1} D_p(f(p,S(p))). \tag{2.23}$$

3

State space formulation of first-order optimization algorithms

We will now consider a general representation of a fairly broad class of first-order optimization methods, and will give a first introduction to the types of optimization algorithms to be considered. Firstly, the state space representation of optimization methods will be introduced. This will be followed by some examples of algorithms on the state space form. Lastly, we will discuss an important property of optimization algorithms, with some necessary conditions for an algorithm on the state space form to be meaningful.

3.1 State space formulation

Consider the minimization problem

$$\min_{y \in \mathbb{R}^p} \sum_{i=1}^{m} f_i(y),$$

for $f_i : \overline{\mathbb{R}^p} \to \mathbb{R}$ and some $p \in \mathbb{N}$. If we let $x_k \in \mathbb{R}^{n \times p}$ denote some auxiliary sequence of n elements in $\mathbb{R}^p$, and let y_k denote the sequence of iterates of the optimization algorithm, then the state space formulation for a first-order optimization algorithm is as follows:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \\ u_k &\in \partial \mathbf{f}(y_k), \end{aligned} \tag{3.1}$$

for $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{m \times n}$ and $D \in \mathbb{R}^{m \times m}$. The number of elements in x, n, is often denoted as lifting. It should be noted that the formulation can be extended to arbitrary Hilbert spaces, but in order to simplify the notation, we will restrict ourselves to $\mathbb{R}^p$. This allows us to deviate from the tensor product notation used in [Upadhyaya et al., 2023]. Instead let $y_k, u_k \in \mathbb{R}^{m\times p}$ and $x_k \in \mathbb{R}^{n\times p}$. Each row in x_k, u_k and y_k represents an element $x_k^{(i)}$, $u_k^{(i)}$ or $y_k^{(i)} \in \mathbb{R}^{1\times p}$. Also, let $\mathbf{f}(y) \in \mathbb{R}^m$ denote the cartesian product of $f_i(y_k^{(i)})$,

$$\mathbf{f}(y_k) = \begin{pmatrix} f_1(y_k^{(1)})) \\ f_2(y_k^{(2)})) \\ \vdots \\ f_m(y_k^{(m)}) \end{pmatrix}. \tag{3.2}$$

The inclusion in (3.1) should be interpreted elementwise as

$$u_k \in \partial\mathbf{f}(y_k) \iff \begin{pmatrix} u_k^{(1)} \in \partial f_1(y_k^{(1)}) \\ u_k^{(2)} \in \partial f_2(y_k^{(2)}) \\ \vdots \\ u_k^{(m)} \in \partial f_m(y_k^{(m)}) \end{pmatrix}. \tag{3.3}$$

The algorithm contains two steps: In each iteration, the following equation has to be solved.

$$y_k = Cx_k + D\partial f(y_k). \tag{3.4}$$

The x_k sequence can then be updated:

$$x_{k+1} = Ax_k + Bu_k. \tag{3.5}$$

For this to be solvable, we require the D matrix to be lower triangular. The values $u_k^{(i)}$ and $y_k^{(i)}$ can then be calculated iteratively, for increasing values of i:

$$y_k^{(i)} - D_{ii}u_k^{(i)} = \sum_{j=1}^{n} C_{ij}x_k^{(j)} + \sum_{j=1}^{i-1} D_{ij}u_k^{(j)}. \tag{3.6}$$

For $D_{ii} = 0$, we calculate $y_k^{(i)}$ directly and from that then compute $u_k^{(i)} = \nabla f(y_k^{(i)})$. Assuming $D_{ii} < 0$, note that the expression (3.6) defines a proximal step according to the optimality condition (**??**). Since the right-hand side only contains known quantities,

$$y_k^{(i)} = \operatorname{prox}_{-f_i D_{ii}}\left(\sum_{j=1}^{n} C_{ij}x_k^{(j)} + \sum_{j=1}^{i-1} D_{ij}u_k^{(j)}\right). \tag{3.7}$$

Each $u_k^{(i)}$ component is uniquely determined from the expression above. If D was not triangular, this interpretation would not work, since the sum over D_{ij} would contain multiple unknown quantities. The overarching algorithm is built on iterates y_k, where $y_k^{(i)}$ is either where the gradient of f_i is evaluated or it is obtained through a proximal evaluation of f_i.

DEFINITION 3.1
We say that $x_\star$, $u_\star$ and $y_\star$ form a fixed-point to the algorithm if

$$\begin{aligned} x_\star &= Ax_\star + Bu_\star \\ y_\star &= Cx_\star + Du_\star \\ u_\star &\in \partial \mathbf{f}(y_\star) \end{aligned}$$

3.2 Examples

In the section that follows, some examples of first-order methods are given and written on state space form. To start, we show the gradient method and then move on to some more interesting examples; the Douglas–Rachford and Davis–Yin operator splitting methods. The term operator splitting method is used to emphasize the fact that they separate the problem of minimizing a sum of functions into proximal evaluations, or gradients, of the individual functions, f_i. These methods are flexible, and due to the use of the proximal operator, they can be used also for non-differentiable functions.

A motivating example, where these kinds of operator-splitting methods are useful is the following problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & x \in C \end{aligned}$$

for a proper, closed convex function $f \in \mathbb{R}^p \to \mathbb{R}$ and a closed convex set C. It is troublesome to use the simple gradient method for this constrained optimization problem, but through a minor reformulation, this problem can be handled well by e.g. the Douglas–Rachford method. The problem can be reformulated as follows:

$$\min_x \quad f(x) + \iota_C(x),$$

using the indicator function, ι, defined in (2.12). This is an unconstrained optimization problem albeit with a non-differentiable objective function. This problem can be solved by many operator-splitting methods, as long as the ι_C function is handled via the proximal operator.

The gradient method

For the gradient method,

$$y_{k+1} = y_k - \gamma \nabla f(y_k), \tag{3.8}$$

the state space formulation becomes

$$\begin{aligned} x_{k+1} &= x_k + (-\gamma)u_k \\ y_k &= x_k + 0 \cdot u_k \\ u_k &= \nabla f(y_k). \end{aligned}$$

The Douglas–Rachford operator splitting method

$$\begin{aligned} y_k^{(1)} &= \operatorname{prox}_{\gamma f_1}(x_k) \\ y_k^{(2)} &= \operatorname{prox}_{\gamma f_2}\left(2y_k^{(1)} - x_k\right) \\ x_{k+1} &= x_k + \lambda(y_k^{(2)} - y_k^{(1)}), \end{aligned}$$

where $\gamma > 0$ and $\lambda \in \mathbb{R}^+$ is a so-called relaxation parameter. Note that the optimality condition (2.9) for the proximal operator can be used to express the following:

$$\begin{aligned} y_k^{(1)} - x_k = -\gamma u_k^{(1)} \quad &\text{for some } u_k^{(1)} \in \partial f_1(y_k^{(1)}) \\ &\implies y_k^{(1)} = x_k - \gamma u_k^{(1)} \end{aligned} \tag{3.9}$$

and

$$\begin{aligned} y_k^{(2)} + x_k - 2y_k^{(1)} = -\gamma u_k^{(2)} \quad &\text{for some } u_k^{(2)} \in \partial f_2(y_k^{(2)}) \\ \implies y_k^{(2)} &= -x_k + 2y_k^{(1)} - \gamma u_k^{(2)} \\ &= x_k - 2\gamma u_k^{(1)} - \gamma u_k^{(2)} \end{aligned} \tag{3.10}$$

Using these expressions for $y_k^{(1)}$ and $y_k^{(2)}$, we obtain the state space formulation

$$\begin{aligned} x_{k+1} &= x_k - \lambda \begin{pmatrix} \gamma & \gamma \end{pmatrix} u_k \\ y_k &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} x_k + \begin{pmatrix} -\gamma & 0 \\ -2\gamma & -\gamma \end{pmatrix} u_k. \end{aligned} \tag{DR}$$

The Davis–Yin three operator splitting method

Another algorithm to consider is the Davis-Yin three-operator splitting method:

$$\begin{aligned} x_k &= \operatorname{prox}_{\gamma f_1}(z_k) \\ z_{k+\frac{1}{2}} &= 2x_k - z_k - \gamma \nabla f_2(x_k) \\ z_{k+1} &= z_k + \lambda(\operatorname{prox}_{\gamma f_3}(z_{k+\frac{1}{2}} - x_k). \end{aligned}$$

with the state space representation

$$\begin{aligned} x_{k+1} &= x_k - \lambda \begin{pmatrix} \gamma & \gamma & \gamma \end{pmatrix} u_k \\ y_k &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} x_k + \begin{pmatrix} -\gamma & 0 & 0 \\ -\gamma & 0 & 0 \\ -2\gamma & -\gamma & -\gamma \end{pmatrix} u_k. \end{aligned} \tag{DY}$$

For a more detailed derivation of the state space formulation, we refer to [Upadhyaya et al., 2023]. Note that $u_k^{(2)}$ is a simple gradient evaluation. In order to assure differentiability we will therefore assume that $f_2 \in \mathcal{F}_{\sigma,\beta}$ for $\beta < \infty$ in all investigated examples.

3.3 The fixed-point encoding property

An important requirement for a proposed algorithm is that solutions to the optimization problem have corresponding fixed points for the algorithm (see definition 3.1) and that it is possible from every fixed point of the algorithm, to extract a solution to the optimization problem. This section will give a more detailed description of the fixed-point encoding property and some necessary conditions.

DEFINITION 3.2
An algorithm on the state space form, (3.1), is said to fulfill the fixed-point encoding property if for every solution to the optimization problem, $y_\star \in \mathbb{R}^{m\times p}$, for which $\sum_{i=1}^m \partial f_i(y_\star) = 0$, there exists $x_\star \in \mathbb{R}^{n\times p}$ and $u_\star \in \mathbb{R}^{m\times p}$ for which $x_\star$, $u_\star$ and $y_\star$ form a fixed point, and where additionally, for all fixed-points of the algorithm it is guaranteed that

$$\sum_{i=1}^m u_\star^{(i)} = 0 \tag{3.11}$$

and

$$y_\star = (y^\star, y^\star, \dots, y^\star)^T, \tag{3.12}$$

for $y^\star \in \mathbb{R}^p$.

The last two conditions ensure the following for a fixed point:

$$0 = \sum_{i=1}^m u_\star^{(i)} \in \sum_{i=1}^m \partial f(y_\star^{(i)}) = \sum_{i=1}^m \partial f(y^\star) \iff y^\star \text{ solves the optimization problem.} \tag{3.13}$$

Those two conditions can succinctly be written

$$\begin{aligned} N^T y_\star &= 0 \\ u_\star = N\hat{u}_\star &\iff u_\star^{(m)} = -\sum_{i=1}^{m-1} u_\star^{(i)} \end{aligned} \tag{3.14}$$

with

$$N = \begin{pmatrix} I \\ -\mathbb{1}^T \end{pmatrix} \in \mathbb{R}^{m\times(m-1)} \tag{3.15}$$

for some

$$\hat{u}_\star = \begin{pmatrix} u_\star^{(1)} \\ u_\star^{(2)} \\ \vdots \\ u_\star^{(m-1)} \end{pmatrix} \in \mathbb{R}^{(m-1)\times p}. \tag{3.16}$$

THEOREM 3.3
An algorithm on state space form (3.1) has the fixed point encoding property if and only if

$$\text{Ran}\begin{pmatrix} BN & 0 \\ DN & 1 \end{pmatrix} \subseteq \text{Ran}\begin{pmatrix} I-A \\ -C \end{pmatrix} \tag{3.17}$$

$$\text{Null}\begin{pmatrix} I-A & -B \end{pmatrix} \subseteq \text{Null}\begin{pmatrix} N^T C & N^T D \\ 0 & \mathbb{1}^T \end{pmatrix}. \tag{3.18}$$

A more detailed account is given in [Upadhyaya et al., 2023], where the theorem is proven.

3.4 Parameterization of first-order methods

We will here make use the fixed point-encoding requirements from (3.17) and (3.18) to find a parameterization of first-order methods with the fixed point encoding property. We will also give a new proof for a known result on the possible dimensions of the x_k sequence in relation to the dimension of the y_k sequence. We use the fact that for matrices $E \in \mathbb{R}^{p\times q}$ and $F \in \mathbb{R}^{p\times r}$

$$\text{Ran}(E) \subseteq \text{Ran}(F) \iff E = FU, \tag{3.19}$$

for some $U \in \mathbb{R}^{r\times q}$. We also use the property

$$\text{Null}(E)^\perp = \text{Ran}(E^T). \tag{3.20}$$

Now, we write

$$\text{Null}(E) \subseteq \text{Null}(F) \iff \text{Ran}(E^T)^\perp \subseteq \text{Ran}(F^T)^\perp \iff \text{Ran}(F^T) \subseteq \text{Ran}(E^T). \tag{3.21}$$

As a consequence, the requirements for the fixed point encoding property (3.17) and (3.18) become

$$\begin{pmatrix} BN & 0 \\ DN & 1 \end{pmatrix} = \begin{pmatrix} I-A \\ -C \end{pmatrix} U \tag{3.22}$$

and

$$\begin{aligned}\begin{pmatrix} C^T N & 0 \\ D^T N & \mathbb{1} \end{pmatrix} &= \begin{pmatrix} I - A^T \\ -B^T \end{pmatrix} V \\ \iff \begin{pmatrix} N^T C & N^T D \\ 0 & \mathbb{1}^T \end{pmatrix} &= V^T \begin{pmatrix} I - A & -B \end{pmatrix} \end{aligned} \tag{3.23}$$

respectively. We will partition $U \in \mathbb{R}^{n\times m}$ and $V \in \mathbb{R}^{n\times m}$ as

$$\begin{aligned} U &= \begin{pmatrix} \tilde{U} & U_m \end{pmatrix} \\ V &= \begin{pmatrix} \tilde{V} & V_m \end{pmatrix}, \end{aligned}$$

where $U_m, V_m \in \mathbb{R}^{n\times 1}$ and $\tilde{U}, \tilde{V} \in \mathbb{R}^{n\times(m-1)}$.

The state-space formulation of the algorithm is not unique. In the following proposition we will show that a change of variables preserves the y_k and the u_k sequences as well as the fixed-point encoding property.

PROPOSITION 3.4
Let $S \in \mathbb{R}^{n\times n}$ be invertible. Given an algorithm of the form (3.1), defined by the matrices A, B, C and D, the change of variable, defined by $\hat{x}_k = S^{-1}x_k$, along with the following altering of the system matrices:

$$\begin{cases} \hat{A} = S^{-1}AS \\ \hat{B} = S^{-1}B \\ \hat{C} = CS \\ \hat{D} = D \end{cases} \tag{3.24}$$

will preserve the y_k-sequence and the original algorithm has the fixed point encoding property if and only if the altered algorithm does.

Proof. First, we show that the new $\hat{x}_k$-sequence is consistent through induction. We can define $\hat{x}_0 = S^{-1}x_0$. Assume now that $\hat{x}_k = S^{-1}x_k$. Then it follows that

$$\hat{x}_{k+1} = \hat{A}\hat{x}_k + \hat{B}u_k = S^{-1}ASS^{-1}x_k + S^{-1}Bu_k = S^{-1}(Ax_k + Bu_k) = S^{-1}x_{k+1}.$$

As a consequence:

$$y_k = Cx_k + Du_k = CSS^{-1}x_k + Du_k = \hat{C}\hat{x}_k + Du_k.$$

It remains to show that the fixed-point encoding property for the altered system iff it holds for the original one:

$$\begin{pmatrix} \hat{B}N & 0 \\ DN & \mathbb{1} \end{pmatrix} = \begin{pmatrix} I - \hat{A} \\ -\hat{C} \end{pmatrix} \hat{U} \iff \begin{pmatrix} S^{-1}BN & 0 \\ DN & \mathbb{1} \end{pmatrix} = \begin{pmatrix} S^{-1}S - S^{-1}AS \\ -CS \end{pmatrix} \hat{U} \iff$$
$$\begin{pmatrix} BN & 0 \\ DN & \mathbb{1} \end{pmatrix} = \begin{pmatrix} I - A \\ -C \end{pmatrix} S\hat{U}_m.$$

By choosing the matrix, $\hat{U}_m = S^{-1}U_m$, we see that the new algorithm fulfills the fixed-point encoding property if and only if the original algorithm does. The second fixed point condition follows similarly. □

3.5 General parameterization for lifting 1

We investigate the case of lifting 1 ($n = 1$). We claim the following general representation and prove it incrementally.

THEOREM 3.5
All state space representable optimization algorithms with the fixed-point encoding property and lifting 1 have the structure

$$x_{k+1} = x_k + \begin{pmatrix} -\beta & -\beta & \dots & -\beta \end{pmatrix} u_k$$

$$y_k = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} x_k + \begin{pmatrix} -\gamma_1 & 0 & 0 & \dots & 0 \\ -\gamma_1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\gamma_1 & 0 & 0 & \dots & 0 \\ -\gamma_1 - \gamma_2 & -\gamma_2 & -\gamma_2 & \dots & -\gamma_2 \end{pmatrix} u_k.$$

PROPOSITION 3.6
For a lifting 1 method, which fulfills the fixed point encoding property, $A = 1$ and B is a vector of constant value , $B = \beta \mathbb{1}^T$ for $\beta \in \mathbb{R}$.

Proof. From the last column in (3.18) we conclude that $V_m^T(I - A) = 0$ and $-V_m B = \mathbb{1}^T$. Since A and V_m are scalars, this can only hold if $A = 1$ and B is the constant row vector $\frac{-1}{V_m}\mathbb{1}^T \equiv \beta \mathbb{1}^T$. □

PROPOSITION 3.7
The C matrix is a constant column matrix and by scaling x we can find an equivalent algorithm, with $C = \mathbb{1}$.

Proof. From the last column in (3.17), it is required that $-CU_m = \mathbb{1}$, which requires $C = -\mathbb{1}/U_m$, for the same reason as above. We can now use the change of variables, $\hat{x}_k = -U_m x_k$. According to proposition 3.4, we obtain the new representation of the algorithm:

$$\hat{C} = -CU_M = -U_m \mathbb{1}/U_m = \mathbb{1}. \tag{3.25}$$

□

Lastly, we have the requirements $DN = -C\tilde{U}$ and $D^T N = -B^T \tilde{V}$. Using the requirement that D is lower-triangular, we write out the products explicitly

$$DN = \begin{pmatrix} d_{11} & 0 & \dots & 0 \\ d_{21} & d_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} - d_{nn} & d_{n2} - d_{nn} & \dots & d_{n(n-1)} - d_{nn} \end{pmatrix}$$

$$C\tilde{U} = \begin{pmatrix} U_1 & U_2 & \dots & U_{m-1} \\ U_1 & U_2 & \dots & U_{m-1} \\ \vdots & \vdots & \dots & \vdots \end{pmatrix}$$

This implies that DN must be constant along the columns, but since DN is lower-triangular, this is only possible if the first column in DN is constant and all other columns are zero. Denoting that constant $-\gamma_1$ and letting d_{nn} be called $-\gamma_2$ we arrive at following theorem,

REMARK 3.8 If $\gamma_1 = \gamma_2$, this is equivalent to the Douglas-Rachford method for $m = 2$ and Davis-Yin for $m = 3$, as seen if compared to (DR) and (DY). If instead $m = 2$ and $\gamma_1 \neq \gamma_2$ this can be shown to be the so-called adaptive Douglas-Rachford method [Dao and Phan, 2019].

COROLLARY 3.9
Any algorithm with dimension $m > 3$ (and $n = 1$), can be reduced to an equivalent algorithm with $m = 3$.

Proof. Since $y^{(1)} = y^{(2)} \dots = y^{(m-1)}$, the algorithm can now be written as

$$\begin{aligned} x_{k+1} &= x_k - \beta \sum_{i=1}^{m} u_k^{(i)} \\ &= x_k - \beta \left(u_k^{(1)} + u_k^{(m)} + \nabla \sum_{i=2}^{m-1} f_i(y_k^{(1)}) \right) \\ y_k^{(1)} &= y_k^{(2)} = \dots = y_k^{(m-1)} = \operatorname{prox}_{\gamma_1 f_1}(x_k) \\ y_k^{(m)} &= \operatorname{prox}_{\gamma_2 f_m} \left(x_k - (\gamma_1 + \gamma_2) u_k^{(1)} - \gamma_2 \sum_{i=2}^{m-1} \nabla f_i(y_k^{(1)}) \right) \\ &= \operatorname{prox}_{\gamma_2 f_m} \left(x_k - (\gamma_1 + \gamma_2) u_k^{(1)} - \gamma_2 \nabla \sum_{i=2}^{m-1} f_i(y_k^{(1)}) \right). \end{aligned}$$

This can be written equivalently using only three functions; f_1, $\sum_{i=2}^{m-1} f_i$ and f_m, with the same x, $y_k^{(1)}$ and $y_k^{(m)}$ sequences.

Since the gradient is linear, there is no difference in calculating $\nabla \sum_{i=2}^{m-1} f_i(y^{(1)})$ or $\sum_{i=2}^{m-1} \nabla f_i(y^{(1)})$ $\square$

4

The performance estimation problem

We will now turn to the performance estimation problem (PEP), the formulation of the worst-case convergence rate of first-order optimization algorithms as a convex optimization problem. Through a series of reformulations, this seemingly very complicated problem can be reduced into a relatively small semidefinite program. In many cases, the result is as exact as the numerical solver. We will start with a general outline of a PEP formulation and will then give a more detailed derivation. We use a simplified version of the problem description in [Upadhyaya et al., 2023].

4.1 PEP outline

For all examples in this report, we will consider the convergence of some sequence, x, to the fixed point corresponding to the optimum, $x_\star$. We can first present an outline of a PEP formulation, inspired by [Drori and Teboulle, 2013], where one iteration of the algorithm is considered:

$$\begin{aligned} &\max_{f, x_0, x_1, x_\star, g_0} \|x_1 - x_\star\|_2^2 \\ &\text{subject to} \\ &\quad f \in \mathcal{F}_{\sigma,\beta} \\ &\quad \|x_0 - x_\star\|_2^2 = R^2 \\ &\quad x_1 = \mathcal{B}(f, x_0, g_0) \\ &\quad g_0 \in \partial f(x_0) \\ &\quad 0 \in \partial f(x_\star). \end{aligned} \tag{4.1}$$

$\mathcal{B}$ describes a step with the specified first-order optimization algorithm and f belongs to the set of β-smooth σ-strongly convex functions, $\mathcal{F}_{\sigma,\beta}$. If we let ρ^2 denote the optimal value derived from this problem, then the optimization problem implies

$$\|x_1 - x_\star\|_2^2 \leq \frac{\rho^2}{R^2}\|x_0 - x_\star\|_2^2. \tag{4.2}$$

In many cases, the objective value, ρ^2 scales linearly with R^2. For that purpose one can let $R^2 = 1$ and obtain the general worst-case convergence rate

$$\|x_1 - x_\star\|_2^2 \leq \rho^2\|x_0 - x_\star\|_2^2. \tag{4.3}$$

It should be noted that it is possible to also consider multiple iterations of the algorithm or use other metrics than the distance to the minimum. An important example of an alternative performance metric, is the function value suboptimality, $f(x_1) - f(x_\star)$.

Convex interpolation

This formulation serves as an initial formulation, however the set of convex functions is infinite-dimensional and this problem is generally non-convex. Some reformulations are needed in order to arrive at a well-behaved optimization problem. The first step is to remove the infinite-dimensional constraints on f. [Drori and Teboulle, 2013] relaxed the problem, replacing the constraints on the function f, with constraints on function values and subgradients only in the points $x_0, \ldots, x_N, x_\star$. In [Taylor et al., 2016], it is shown that this can be done equivalently, as long as the constraints are formulated in a certain way. For the intended purpose of rewriting the PEP into a small optimization problem, this does exactly what is needed.

THEOREM 4.1—TAYLOR
There exists a β-smooth, σ-strongly convex function, f, with function values $f(x_i) = f_i$ and subgradients $g_i \in \partial f(x_i)$ if and only if there are triplets (x_i, f_i, g_i) for $i \in \mathcal{I}$ which fulfill the inequalities

$$f_i \geq f_j + g_j^T(f_i - f_j) + \frac{1}{2(1-\frac{\sigma}{\beta})}\left(\frac{1}{\beta}||g_i - g_j||_2^2 + \sigma||x_i - x_j||_2^2 - 2\frac{\sigma}{\beta}(g_j - g_i)^T(x_j - x_i)\right) \tag{4.4}$$

for all $j \in \mathcal{I}$. This theorem shows that the infinite-dimensional convexity constraint can be relaxed into equivalent, finite-dimensional constraints.

4.2 State space formulation of the PEP

The following PEP formulation will be based on the formulation used in [Upadhyaya et al., 2023]. We will follow their outline, with some slight modifications and simplifications. We can construct this formulation from the base formulation of the performance estimation problem, however since we consider one iteration of an optimization algorithm on state space form, we have to optimize over points x_i, y_i, f_i and u_i for $i \in \{0, 1, \star\}$, with the additional constraints that they fulfill the state-space equations. We will only consider the quadratic objective function

$$\sum_{l=1}^{n} ||x_i^{(l)} - x_\star^{(l)}||_2^2 = \mathrm{Tr}((x_i - x_\star)^T (x_i - x_\star)), \tag{4.5}$$

but since $n = 1$ for all cases in this report we only need consider $||x_i - x_\star||_2^2$ and search for the linear convergence rate ρ for which

$$||x_1 - x_\star||_2^2 \leq \rho^2 ||x_0 - x_\star||_2^2. \tag{4.6}$$

Note that more general cases can be considered. The general quadratic objective functions from [Upadhyaya et al., 2023], can be incorporated by instead finding a linear convergence rate for

$$\mathrm{Tr}(\zeta^T C \zeta) + q^T (f(y_i) - f(y_\star)), \tag{4.7}$$

for a positive semidefinite matrix C and $\zeta = \begin{pmatrix} x_k - x_\star \\ u_0 \\ u_1 \\ \hat{u}_\star \end{pmatrix}$. This combines function value suboptimality, scaled distances to the fixed points and subdifferentials. It is possible to prove that the objective value at the optimum scales linearly with $||x_0 - x_\star|| = R^2$. It holds as a consequence that

$$\rho^2 = \max \frac{||x_1 - x_\star||_2^2}{||x_0 - x_\star||_2^2} = \max_{||x_0 - x_\star||_2^2 = 1} ||x_1 - x_\star||_2^2 \tag{4.8}$$

and we will optimize $||x_1 - x_\star||_2^2$ under the constraint that $||x_0 - x_\star||_2^2 = 1$.

4.3 Initial formulation

We start with the following, needlessly verbose problem

$$\begin{aligned} \sup_{x_0, x_1, x_\star, f, u_0, u_1, u_\star} \quad & \|x_1 - x_\star\|_2^2 \\ \text{subject to } & \|x_0 - x_\star\|_2^2 = 1 \\ & y_0 = Cx_0 + Du_0 \\ & x_1 = Ax_0 + Bu_0 \\ & y_1 = Cx_1 + Du_1 \\ & x_\star = Ax_\star + Bu_\star \\ & y_\star = Cx_\star + Du_\star \\ & u_k = \partial \mathbf{f}(y_k) \\ & f_i \in \mathcal{F}_{\sigma_i, \beta_i}. \end{aligned} \tag{4.9}$$

Similarly to the previous base formulation, this is not a convex optimization problem, and it is more complex than need be. It will therefore be reformulated, step by step. Many of the variables to be optimized over, can be rewritten in terms of $x_0, x_\star, u_0, u_1$ and $u_\star$. We will also replace the convexity constraint through theorem 4.1.

For a fixed point, we require that the sum of $u_\star^{(i)}$ to be zero. Just like in the previous chapter, this means the last $u_\star$ component is uniquely determined by the rest and we can instead optimize over $\hat{u}_\star \in \mathbb{R}^{(m-1)\times p}$, letting

$$u_\star = N\hat{u}_\star = \begin{pmatrix} \hat{u}_\star \\ -\mathbb{1}^T \hat{u}_\star \end{pmatrix}, \tag{4.10}$$

for N defined in (3.15). Let us introduce

$$\zeta = \begin{pmatrix} x_0 - x_\star \\ u_0 \\ u_1 \\ \hat{u}_\star \end{pmatrix} \in \mathbb{R}^{(n+3m-1)\times p} \tag{4.11}$$

and note that most expressions in the problem (4.9) can be expressed only through the elements in ζ. We firstly express $x_1 - x_\star$ and $x_0 - x_\star$ in terms of ζ,

$$x_0 - x_\star = \begin{pmatrix} I & 0 & 0 & 0 \end{pmatrix} \zeta \equiv \Sigma_0 \zeta. \tag{4.12}$$

Secondly,

$$x_1 - x_\star = Ax_0 + Bu_0 - (Ax_\star + BN\hat{u}_\star) = A(x_0 - x_\star) + Bu_0 - BN\hat{u}_\star \tag{4.13}$$

and consequently

$$\begin{pmatrix} x_1 - x_\star \end{pmatrix} = \begin{pmatrix} A & B & 0 & -BN \end{pmatrix} \zeta \equiv \Sigma_1 \zeta \tag{4.14}$$

No more expressions depend on x_1 and we obtain the following intermediary optimization problem

$$\begin{aligned}
\sup_{x_0, x_1, x_\star, f, u_0, u_1, u_\star} \quad & \operatorname{Tr}\left(\zeta^T \Sigma_1^T \Sigma_1 \zeta\right) \\
\text{subject to} \quad & \operatorname{Tr}\left(\zeta^T \Sigma_0^T \Sigma_0 \zeta\right) = 1 \\
& y_0 = Cx_0 + Du_0 \\
& y_1 = C(Ax_0 + Bu_0) + Du_1 \\
& x_\star = Ax_\star + Bu_\star \\
& y_\star = Cx_\star + Du_\star \\
& u_k = \partial \mathbf{f}(y_k) \\
& f_i \in \mathcal{F}_{\sigma_i, \beta_i}
\end{aligned} \tag{4.15}$$

since

$$\sum_{i=1^n} \|x_1^{(i)} - x_\star^{(i)}\|_2^2 = \operatorname{Tr}\left((\Sigma_1 \zeta)^T (\Sigma_1 \zeta)\right) = \operatorname{Tr}\left(\zeta^T \Sigma_1^T \Sigma_1 \zeta\right). \tag{4.16}$$

4.4 Convexity constraints

As an extension of theorem 4.1 from [Taylor et al., 2016], a suitable reformulation is made in [Upadhyaya et al., 2023]. Let the column vector $e_l \in \mathbb{R}^m$ denote the the l:th canonical basis vector in R^m and let the matrix M_l be defined through the Kroenecker product

$$M_l = \begin{cases} \frac{1}{2(\beta_l - \sigma_l)} \begin{pmatrix} \beta_l \sigma_l & -\sigma_l & \beta_l \\ -\sigma_l & 1 & -1 \\ \beta_l & -1 & 1 \end{pmatrix} \otimes \operatorname{diag}(e_l) & \text{if } \beta_l < \infty \\ \frac{1}{2} \begin{pmatrix} \sigma_l & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \otimes \operatorname{diag}(e_l) & \text{otherwise.} \end{cases} \tag{4.17}$$

Let also ξ denote the matrix

$$\xi_{ij} = \begin{pmatrix} y_i - y_j \\ u_i \\ u_j \end{pmatrix} \quad \text{for } i \neq j \in \{0, 1, \star\} \tag{4.18}$$

The convexity constraints from [Upadhyaya et al., 2023], can be written

$$-e_l^T(f_i - f_j) + \operatorname{Tr}\left(\xi_{ij}^T M_l \xi_{ij}\right) \leq 0 \tag{4.19}$$

for all $i \neq j \in \{0, 1, \star\}$ and $l \in [0, 1, \dots, m]$.

The next method of simplifying the expression is to describe all instances of, y_0 and y_1 in terms of x_0, u_0 and u_1. The inequalities representing the convexity constraints now become more complex, but we can drop almost all equality constraints. We arrive at an optimization over $u_0, u_1, \hat{u}_\star, x_0 - x_\star$. All other variables are uniquely determined from these. We find expressions on the form

$$-e_l^T(F_i - F_j) + \operatorname{Tr}(\xi_{ij}^T M_l \xi_{ij}), \tag{4.20}$$

where the $y_i - y_j$ terms can be written in terms of $x_0 - x_\star$ since

$$\begin{aligned} y_1 - y_0 &= Cx_1 + Du_1 - Cx_0 - Du_0 = C(Ax_0 + Bu_0) + Du_1 - Cx_\star + Cx_\star - Cx_0 - Du_0 \\ &= CAx_0 + CBu_0 + Du_1 - C(Ax_\star + Bu_\star) - C(x_0 - x_\star) - Du_0 \\ &= C(A - I)(x_0 - x_\star) + CBu_0 + D(u_1 - u_0) \\ &= \begin{pmatrix} C(A-I) & CB - D & D & 0 \end{pmatrix} \zeta \end{aligned} \tag{4.21}$$

Similarly, expressions can be derived for $y_0 - y_\star$ and $y_1 - y_\star$:

$$\begin{aligned} y_0 - y_\star &= Cx_0 + Du_0 - Cx_\star - Du_\star = \begin{pmatrix} C & D & 0 & -DN \end{pmatrix} \zeta \\ y_1 - y_\star &= Cx_1 + Du_1 - Cx_\star - Du_\star = C(Ax_0 + Bu_0) + Du_1 - C(Ax_\star + Bu_\star) - Du_\star \\ &= \begin{pmatrix} CA & CB & D & -CBN - DN \end{pmatrix} \zeta \end{aligned} \tag{4.22}$$

and now, since there is no dependence left on $y_\star$, y_1 or x_1, all equality constraints for these can be dropped. We use these expressions to rewrite the vector

$$\begin{aligned} \begin{pmatrix} y_1 - y_0 \\ u_1 \\ u_0 \end{pmatrix} &= \begin{pmatrix} C(A-I) & CB - D & D & 0 \\ 0 & 0 & I & 0 \\ 0 & I & 0 & 0 \end{pmatrix} \zeta \equiv E_{10}\zeta \\ \begin{pmatrix} y_0 - y_\star \\ u_1 \\ \hat{u}_\star \end{pmatrix} &= \begin{pmatrix} C & D & 0 & -DN \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & N \end{pmatrix} \zeta \equiv E_{0\star}\zeta \\ \begin{pmatrix} y_1 - y_\star \\ u_1 \\ \hat{u}_\star \end{pmatrix} &= \begin{pmatrix} CA & CB & D & -CBN - DN \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & N \end{pmatrix} \zeta \equiv E_{1\star}\zeta \end{aligned} \tag{4.23}$$

with the first block-row of E_{ji} having opposite sign from the corresponding rows in E_{ij} and the remaining rows staying the same. We can now write the quadratic expressions in terms of ζ

$$\operatorname{Tr}(\xi_{ij}^T M_l \xi_{ij}) = \operatorname{Tr}(\zeta^T E_{ij}^T M_l E_{ij} \zeta) \tag{4.24}$$

and now arrive at the simplified problem

$$\begin{aligned}\max_{\zeta, f_0, f_1, f_\star} \quad & \mathrm{Tr}(\zeta^T \Sigma_1^T \Sigma_1 \zeta) \\ & \mathrm{Tr}(\zeta^T \Sigma_0^T \Sigma_0 \zeta) \\ & -e_l^T (f_i - f_j) + \mathrm{Tr}(\zeta_l^T E_{ij}^T M_l E_{ij} \zeta_l), \leq 0 \quad \forall i, j, l \\ & x_\star = Ax_\star + Bu_\star. \end{aligned} \tag{4.25}$$

However, from the first fixed-point encoding assumption in (3.17), we know that always there must exist a point $x_\star = Ax_\star + Bu_\star = Ax_\star + BN\hat{u}_\star$ since

$$\mathrm{Ran}(BN) \subseteq \mathrm{Ran}(I - A) \iff \exists U : BN = (I - A)U \implies BN\hat{u}_\star = (I - A)U\hat{u}_\star \equiv (I - A)x_\star. \tag{4.26}$$

Since there is no other direct dependence on $x_\star$, this expression can now be dropped.

Gramian formulation

Lastly, we rewrite the quadratic expressions using the Gramian, which can be defined as $G = \zeta\zeta^T$. This is done as a final step, to arrive at a semidefinite program. Recall that

$$\zeta = \begin{pmatrix} x_0 - x_\star \\ u_0 \\ u_1 \\ \hat{u}_\star \end{pmatrix}. \tag{4.27}$$

Since we can write the quadratic expression $x^T G x$ as $(\zeta^T x)^T \zeta^T x \geq 0$ we know this matrix is symmetric and positive semidefinite. If we now let ζ_i denote the i:th row in ζ, note that the quadratic functions can be rewritten

$$\mathrm{Tr}(\zeta^T C \zeta) = \mathrm{Tr}(C\zeta\zeta^T) = \mathrm{Tr}(CG(\zeta)) = \langle C, G \rangle, \tag{4.28}$$

where the first equality follows from the identitity $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$. Lastly, we note that all function values arise as a difference. $f(y_1) - f(y_\star) \equiv \Delta f_1$, $f(y_0) - f(y_\star) \equiv \Delta f_0$ and $\Delta f_\star \equiv 0$ and all terms involving f can be replaced with theses expressions. We finally arrive at a Semidefinite optimization problem over the Gramian matrix,

$$\begin{aligned}\sup_{G, \Delta f_0, \Delta f_1} \quad & \langle \Sigma_1^T \Sigma_1, G \rangle \\ \text{s.t} \quad & \langle \Sigma_0^T \Sigma_0, G \rangle = 1 \\ & -e_l^T (\Delta f_i - \Delta f_j) + \langle E_{ij}^T M_l E_{ij}, G \rangle \leq 0 \quad \forall i \neq j \in \{0, 1, \star\} \text{ and } \forall l \in [0, 1, \dots, m] \\ & G \succeq 0, \end{aligned} \tag{PEP}$$

but the question arises; is this problem equivalent to the original one? From a given vector ζ, we can always obtain a Gramian but can we always construct a vector ζ from a given positive semidefinite matrix G? As mentioned in [Taylor et al., 2016] and [Upadhyaya et al., 2023], this is dimension dependent. Since G is positive semidefinite we can obtain an admissible P matrix from the Cholesky decomposition of G. However, if we also have some constraint on the dimension, p, for $f : \mathbb{R}^p \longmapsto \mathbb{R}$, then the PEP-problem is inexact for $p < n+3m-1$. As a way to view this, consider what happens if the Gramian, G has rank more than p. Then we cannot obtain a factorization $\zeta\zeta^T$ where ζ is a $n+3m-1 \times p$ matrix, since the product $\zeta\zeta^T$ can have rank at most p. This is not a problem for $p \geq n+3m-1$, but when this problem occurs, the problem (PEP) gives an upper bound on the worst-case convergence rate.

4.5 The PEP formulation of a few algorithms

We will give a few examples, in order to give a little more intutition for the gramian PEP formulation, using the same algorithms as in section 3.

Gradient Method

From before, we have

$$\begin{cases} \Sigma_1 = \begin{pmatrix} A & B & 0 & -BN \end{pmatrix} \\ \Sigma_0 = \begin{pmatrix} I & 0 & 0 & 0. \end{pmatrix} \end{cases}$$

For the gradient method, the resulting Σ_1 matrix is

$$\Sigma_1 = \begin{pmatrix} A & B & 0 \end{pmatrix} = \begin{pmatrix} 1 & -\gamma & 0 \end{pmatrix}, \tag{4.29}$$

since $N \in \mathbb{R}^{1\times 0}$ disappears due to the dimensions. The products of the Σ matrices are

$$\begin{cases} \Sigma_1^T\Sigma_1 = \begin{pmatrix} 1 & -\gamma & 0 \\ -\gamma & \gamma^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \Sigma_0^T\Sigma_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{cases}$$

and assuming the function $f \in \mathbb{R}^p$ is allowed dimension p at least $n+3m-1 = 3$, the gramian matrix can be interpreted as

$$G = \begin{pmatrix} \|x_0 - x_\star\|^2 & (x_0 - x_\star)^T u_0 & (x_0 - x_\star)^T u_1 \\ u_0^T(x_0 - x_\star) & \|u_0\|_2^2 & u_0^T u_1 \\ u_1^T(x_0 - x_\star) & u_1^T u_0 & \|u_1\|_2^2 \end{pmatrix} \tag{4.30}$$

We can use these matrices to show that the trace formulation gives us expressions in correspondence to the original formulation

$$\begin{aligned} \operatorname{Tr}(\Sigma_0^T \Sigma_0) = 1 &\iff \operatorname{Tr}\left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} G \right) = 1 \\ &\iff \|x_0 - x_\star\|^2 = 1. \end{aligned} \tag{4.31}$$

Note that the objective function depends on the chosen algorithm and from the following calculations, we show how the gramian formulation corresponds to the original formulation:

$$\begin{aligned} \operatorname{Tr}\left(\begin{pmatrix} 1 & -\gamma & 0 \\ -\gamma & \gamma^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} G \right) &= \|x_0 - x_\star\|_2^2 - 2\gamma(x_0 - x_\star)^T u_0 + \gamma^2 \|u_0\|_2^2 \\ &= \|(x_0 - \gamma u_0) - x_\star\|_2^2 = \|x_1 - x_\star\|_2^2. \end{aligned} \tag{4.32}$$

The Douglas–Rachford method

For the Douglas-Rachford method, a similar expression can be obtained, however since $m = 2$, the dimensions are larger

$$\begin{cases} \Sigma_1 = \begin{pmatrix} A & B & 0 & -BN \end{pmatrix} = \begin{pmatrix} 1 & -\lambda\gamma & -\lambda\gamma & 0 & 0 & 0 \end{pmatrix} \\ \Sigma_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \end{cases} \tag{4.33}$$

5

Differentiating a convex optimization problem

The following chapter will describe the theory of differentiation of conic optimization problems as presented in [Agrawal et al., 2020] and [Busseti et al., 2018]. The more general optimization results can be read in [Boyd and Vandenberghe, 2011]. We will start by showing some fundamental principles of conic optimization, follow up by introducing some necessary concepts for the differentiation and lastly combine all results in order to show how the full differentiation can be performed. The convex optimization problem we ultimately want to differentiate is the PEP as described in section 4. This chapter will be followed by a chapter on the PEP-specific aspects of differentiation.

5.1 Conic programming

Throughout, we will consider convex optimization problems of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\, s\in\mathbb{R}^m} \quad & c^T x \\ \text{subject to} \quad & s = b - \mathcal{A}x \\ & s \in \mathcal{K}, \end{aligned} \tag{P}$$

where $\mathcal{K} \subseteq \mathbb{R}^m$ denotes a convex cone, $\mathcal{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$.This class of problems is called conic optimization problems, or conic programming problems. The dual to the conic optimization problem (P) is of the form

$$\begin{aligned} \min_{y\in\mathbb{R}^m} \quad & b^T y \\ \text{subject to} \quad & \mathcal{A}^T y + c = 0 \\ & y \in \mathcal{K}^*. \end{aligned} \tag{D}$$

A detailed derivation can be read in [Boyd and Vandenberghe, 2011].

PROPOSITION 5.1
For points $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ that are feasible for the primal (P) and dual problems (D) respectively, the following holds for the objective functions of the primal and dual problems

$$c^T x + b^T y \geq 0. \tag{5.1}$$

Proof. This follows from the definition of the dual cone. We know that $c = -\mathcal{A}^T y$ and that $b = s + \mathcal{A}x$, which implies that

$$c^T x + b^T y = -y^T \mathcal{A}x + s^T y + x^T \mathcal{A}^T y = s^T y \geq 0, \tag{5.2}$$

since $s \in \mathcal{K}$ and $y \in \mathcal{K}^*$. □

For the optimal points for the primal and dual problems, respectively, this sum is often called the duality gap. For the purposes of this report, we will be interested in the dual problem and we state the following important result. Under relatively mild assumptions, this duality gap can be assumed to be zero (see for example Slater's condition in [Boyd and Vandenberghe, 2011]).

PROPOSITION 5.2
Let $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$ and $y \in \mathbb{R}^m$ be feasible, i.e.

$$\begin{cases} s = b - \mathcal{A}x \in \mathcal{K} \\ y \in \mathcal{K}^* \\ \mathcal{A}^T y + c = 0, \end{cases} \tag{5.3}$$

and let also $s^T y = 0$. Then (x, s) solves the primal problem (P) and y solves the dual problem (D).

Proof. From 5.2 and the assumption that $s^T y = 0$, we conclude that

$$0 = s^T y = b^T y + c^T x. \tag{5.4}$$

Assume furthermore that (x, s) is not optimal, i.e that there exists an admissible point $(\hat{x}, \hat{s}) \in \mathbb{R}^n \times \mathcal{K}$ for which $\hat{s} = b - \mathcal{A}\hat{x}$ and where $c^T \hat{x} < c^T x$. Then

$$b^T y + c^T \hat{x} < b^T y + c^T x = 0, \tag{5.5}$$

which is impossible, according to (5.2).

Similarly, if we assume there exists an admissible point $\hat{y} \in \mathcal{K}^*$ for which $\mathcal{A}^T \hat{y} + c = 0$ and $b^T \hat{y} < b^T y$, we again arrive at a contradiction because

$$c^T x + b^T \hat{y} < c^T x + b^T y = 0. \tag{5.6}$$

□

5.2 Solution mapping for convex optimization problems

In order to differentiate a conic optimization problem we need to define a mapping from the problem parameters, $\mathcal{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ to the primal and dual solutions of interest, $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$ and $y \in \mathbb{R}^m$. The backbone of this mapping is the homogeneous self-dual embedding as described in [Busseti et al., 2018]. It acts as a certificate of either optimality (or infeasibility) and therefore pairs well with the implicit function theorem. For the full solution mapping, we consider the following formulation:

$$(x,y,s) = \mathcal{S}(\mathcal{A},b,c) = \phi \circ S \circ Q(\mathcal{A},b,c) \tag{5.7}$$

The function

$$Q(\mathcal{A},b,c) : \mathbb{R}^{m\times n} \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^{(n+m+1)\times(n+m+1)} \tag{5.8}$$

simply constructs a matrix, Q, containing all problem parameters, $\mathcal{A}$, b and c. Secondly, the function

$$S(Q) : \mathbb{R}^{(n+m+1)\times(n+m+1)} \to \mathbb{R}^{n+m+1} \tag{5.9}$$

can be defined from the implicit function theorem in combination with the self-dual embedding, it and maps the matrix Q to a solution, z, of the self-dual embedding. We will never find a closed-form expression for the function $S(Q)$, however the implicit function theorem can be used to prove that the function exists and gives us an expression for the derivative of $S(Q)$. The function

$$\phi(z) : \mathbb{R}^{n+m+1} \to \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \tag{5.10}$$

lastly maps z to a solution (x,y,s).

5.3 Self-dual embedding

In the section that follows, we will describe the self-dual embedding and show how to construct the $Q(\mathcal{A},b,c)$ and $\phi(z)$ functions, and follow up by describing how the implicit function theorem can be used to differentiate the function $S(Q)$.

Let us now define the skew-symmetric matrix

$$Q(\mathcal{A},b,c) = \begin{pmatrix} 0 & \mathcal{A}^T & c \\ -\mathcal{A} & 0 & b \\ -c^T & -b^T & 0. \end{pmatrix} \in \mathbb{R}^{(n+m+1)\times(n+m+1)} \tag{5.11}$$

and consider vectors $u \in \mathbb{R}^{n+m+1}$ and $v \in \mathbb{R}^{n+m+1}$ that satisfy

$$\begin{cases} u = (u_1, u_2, \tau) \in \mathbf{K} = R^n \times \mathcal{K}^* \times R_+ \\ v = (0, v_2, \kappa) \in \mathbf{K}^* = R^m \times \mathcal{K} \times R_+ \\ Qu = v \end{cases} \tag{Self-dual embedding}$$

and where at least one of κ and τ is strictly positive. These vectors, u and v, along with the matrix Q, define the self-dual embedding.

PROPOSITION 5.3
If $u \in \mathbf{K}$ and $v \in \mathbf{K}^*$ are solutions to the self-dual embedding, $Qu = v$, then either τ or κ is zero.

Proof. Since $Qu = v$ and Q is skew-symmetric, we can show that $u^T v = u^T Qu$ is zero according to proposition (2.16). Writing out the inner product $u^T v$ we conclude that

$$0 = u^T v = 0 + u_2^T v_2 + \tau\kappa. \tag{5.12}$$

However, since $u_2 \in \mathcal{K}^*$ and $v_2 \in \mathcal{K}$ it holds that $u_2^T v_2 \geq 0$ by the definition of the dual cone, and since κ and τ are non-negative, we can only have equality if κ or τ is zero and $u_2^T v_2 = 0$. □

This gives three cases, $\tau > 0$ and $\kappa = 0$, $\tau = 0$ and $\kappa > 0$, and $\tau = 0$ and $\kappa = 0$. Let us look into the $\tau > 0$ case.

PROPOSITION 5.4
For $\tau > 0$ and $\kappa = 0$, we can derive optimal solutions to the primal and dual problems:

$$\begin{cases} x = \frac{u_1}{\tau} \\ y = \frac{u_2}{\tau} \\ s = \frac{v_2}{\tau} \end{cases} \tag{5.13}$$

Proof. First note that $s = \frac{v_2}{\tau} \in \mathcal{K}$ and $y = \frac{u_2}{\tau} \in \mathcal{K}^*$. According to the definition of the self-dual embedding,

$$Qu = v \implies \begin{pmatrix} \mathcal{A}^T u_2 + c\tau \\ -\mathcal{A}u_1 + b\tau \\ -c^T u_1 - b^T u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ v_2 \\ \kappa \end{pmatrix}. \tag{5.14}$$

From the first equality, we find that

$$0 = \mathcal{A}^T u_2 + c\tau = (\mathcal{A}^T y + c)\tau, \tag{5.15}$$

meaning y is admissible for the dual problem. The second equality shows us that x and s are admissible for the primal problem

$$-\mathcal{A}x\tau + b\tau = s\tau \tag{5.16}$$

and from the last equation $-c^T x\tau - b^T y\tau = 0$, we can use (5.2) to prove optimality. □

For $\kappa > 0$, it is possible to prove that either the primal or the dual problem is infeasible, using a similar argument.

5.4 Differentiating the residual mapping

The previous section gives an overview of the self-dual embedding and how to construct a solution (x,y,s) from solution vectors $u \in \mathbf{K}$ and $v \in \mathbf{K}^*$. For the sake of differentiating the conic problem, one piece is still missing: we still have not found a derivate for the function $S(Q)$ or even shown that this function exists. The main idea is to define the residual function, $R(z) = Qu(z) - v(z)$, and normalize it. This serves as a measure of how close u and v are to solving the self-dual embedding. This function is differentiable (almost everywhere) and can ultimately be used together with the implicit function theorem, in order to prove the existence and define the derivative of the function $S(Q)$.

From the self-dual embedding, we define the conic complementary set as

$$\mathcal{C} = \{(u,v) \in \mathbf{K} \times \mathbf{K}^* \mid u^T v = 0\}. \tag{5.17}$$

Let

$$\Pi_{\mathbf{K}} : \mathbb{R}^{n+m+1} \to \mathbf{K} \subset \mathbb{R}^{n+m+1} \tag{5.18}$$

denote orthogonal projection onto the set $\mathbf{K}$, and let

$$\Pi^*_{-\mathbf{K}^*} : \mathbb{R}^{n+m+1} \to -\mathbf{K}^* \subseteq \mathbb{R}^{n+m+1} \tag{5.19}$$

denote projection onto $-\mathbf{K}^*$. It follows from (2.14), that $\Pi_{-\mathbf{K}^*} = I - \Pi_{\mathbf{K}}$ and that $\Pi_{\mathbf{K}}(z)$ is orthogonal to $\Pi_{-\mathbf{K}^*}(z)$. For any vector z, we can now introduce the Minty Parameterization, $M : \mathbb{R}^n \to \mathcal{C}$ defined as

$$M(z) = (\Pi_{\mathbf{K}}(z), -\Pi_{-\mathbf{K}^*}(z)), \tag{5.20}$$

with the purpose of uniquely mapping a vector $z \in \mathbb{R}^{n+m+1}$ onto elements $(u,v) \in \mathcal{C}$, which will serve as solution candidates for the self-dual embedding. We say that z is a solution to the self-dual embedding if $(u,v) = M(z)$ solve the self-dual embedding.

PROPOSITION 5.5
M is bijective with inverse $M^{-1}(u,v) = u - v$.

Proof. It follows from the definition of the Minty parameterization and the Moreau identity (2.11) that

$$M^{-1}(M(z)) = \Pi_{\mathbf{K}}(z) - (-\Pi_{-\mathbf{K}^*}(z)) = \Pi_{\mathbf{K}}(z) + \Pi_{-\mathbf{K}^*}(z) = z \quad \forall z \in \mathbb{R}^{n+m+1}. \tag{5.21}$$

□

Since we obtain candidate vectors $u \in \mathbf{K}$ and $v \in \mathbf{K}^*$, from the Minty parametrization, the conditions of the self-dual embedding, $Qu = v$ can now be written in terms of z:

$$Q\Pi_{\mathbf{K}}(z) = -\Pi_{-\mathbf{K}^*}(z). \tag{5.22}$$

Let the residual function be defined as

$$R(z,Q) = Q\Pi_{\mathbf{K}}(z) + \Pi_{-\mathbf{K}^*}(z). \tag{5.23}$$

This is zero precisely when $Qu = v$ for $(u,v) = M(z)$, i.e. when z is a solution to the self-dual embedding described above.

PROPOSITION 5.6
The derivatives of $R(z,Q)$

$$\begin{aligned} D_Q R(z,Q)U &= U\Pi_{\mathbf{K}}(z) \\ D_z R(z,Q) &= QD\Pi_{\mathbf{K}}(z) + D\Pi_{-\mathbf{K}^*}(z), \end{aligned} \tag{5.24}$$

for some matrix U, as long as the derivatives of the projections are defined.

COROLLARY 5.7
Since both $\Pi_{\mathbf{K}}$ and $\Pi_{\mathbf{K}^*}(z)$ are Lipschitz continuous, it follows from Rademacher's theorem (2.17) that $R(z,Q)$ is differentiable almost everywhere.

5.5 Normalized residual mapping

Now we turn to the normalized residual mapping. To see why this is necessary, note that the self-dual embedding is invariant to scaling, i.e. if $u \in \mathbf{K}$ and $v \in \mathbf{K}^*$ solve the embedding then so do λu and λv for $\lambda \in \mathbb{R}$. If u and v are not solutions however, then the residual $Qu - v$ can be made arbitrarily close to zero by scaling u and v by the constant constant λ. For that reason, we normalize so the last element in z is always 1. The normalized residual mapping is defined as

$$N(z,Q) = \frac{R(z,Q)}{z_{n+m+1}} = \frac{R(z,Q)}{e^T z}, \tag{5.25}$$

where z_{n+m+1} can be written as $e^T z$, if by e, we denote the last unit vector, $e = (0,\dots,0,1)^T$. Whenever $R(z,Q)$ is differentiable, so is $N(z,Q)$, and we obtain the following expression

$$\begin{aligned} \mathrm{D}_Q N(z,Q)U &= \mathrm{D}_Q \frac{R(z,Q)}{z_{n+m+1}} U = U \frac{\Pi_{\mathbf{K}}}{z_{n+m+1}} \\ \mathrm{D}_z N(z,Q) &= \frac{\mathrm{D}R(z,Q)}{z_{n+m+1}} - \frac{R(z,Q)}{z_{n+m+1}^2} e^T. \end{aligned} \tag{5.26}$$

The second term vanishes if z solves the self-dual embedding, and the result is as follows:

$$\mathrm{D}_z N(z,Q) = \frac{\mathrm{D}R(z,Q)}{z_{n+m+1}} = ((Q-I)D\Pi(z) + I)/z_{n+m+1}. \tag{5.27}$$

Now the function, $\phi(z) : \mathbb{R}^{n+m+1} \to \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m$, which constructs a solution to the optimization problem fromt z, can be defined. It is already established in (5.13) that $(x,y,s) = (\frac{u_1}{\tau}, \frac{u_2}{\tau}, \frac{v_2}{\tau})$ is a solution. If we make the partition $z = (z_1, z_2, z_3) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}$ and investigate the projections onto $\mathbf{K}$, we find the following:

$$u = \Pi_{\mathbf{K}}(z) = \Pi_{\mathbb{R}}(z_1) \times \Pi_{\mathcal{K}^*}(z_2) \times \Pi_{\mathbb{R}^+}(z_3) = z_1 \times \Pi_{\mathcal{K}^*}(z_2) \times \tau, \tag{5.28}$$

where we assume $z_3 = \tau \geq 0$. Moreover,

$$v = -\Pi_{-\mathbf{K}^*} = -(z - \Pi_{\mathbf{K}}(z)) = u - z = 0 \times \Pi_{\mathcal{K}^*}(z_2) - z_2 \times 0. \tag{5.29}$$

Ultimately, the solution $(x,y,s) = (\frac{u_1}{\tau}, \frac{u_2}{\tau}, \frac{v_2}{\tau})$ can be written

$$(x,y,s) = \phi(z) = (z_1, \Pi_{\mathcal{K}^*}(z_2), \Pi_{\mathcal{K}^*}(z_2) - z_2)/z_3 \tag{5.30}$$

and after normalization, where z_3 is assumed to be 1, the derivative is

$$\mathrm{D}\phi(z) = \begin{pmatrix} I & 0 & -x \\ 0 & \mathrm{D}\Pi_{\mathcal{K}^*}(z_2) & -y \\ 0 & \mathrm{D}\Pi_{\mathcal{K}^*}(z_2) - I & -s \end{pmatrix}. \tag{5.31}$$

A solution z can similarly be constructed from (x,y,s)

$$(z_1, z_2, z_3) = (x, y - s, 1). \tag{5.32}$$

5.6 Full solution mapping

Lastly, the solution mapping $S(Q)$ needs to be differentiated. After that, we have all necessary expressions to differentiate a conic program.

If we let the the normalized residual mapping, $N(z,Q)$, be used as the function f in the the implicit function theorem (theorem 2.18), it follows as a consequence that there exists a solution mapping $S(Q) : \mathbb{R}^{(n+m+1)\times(n+m+1)} \to \mathbb{R}^{n+m+1}$, whenever $N(z,Q)$ is differentiable and non-singular. The solution mapping function, $S(Q)$, maps the problem parameters, Q, to a solution of the self-dual embedding, denoted z, in a neighborhood of a solution , $\bar{z}$, $\bar{Q}$ for which $N(\bar{z}, \bar{Q}) = 0$. If N is non-singularly differentiable, the theorem states that (see (2.23)) the derivative is

$$\mathrm{D}\mathcal{S}(Q) = -\left(\mathrm{D}N_z\left(s(Q), Q\right)\right)^{-1} \mathrm{D}N_Q\left(s(Q), Q\right). \tag{5.33}$$

From the expressions (5.26), the closed form expression is

$$\mathrm{D}\mathcal{S}(Q) = -((Q - I)\mathrm{D}\Pi(z) + I)^{-1}\mathrm{D}N_Q(S(Q), Q) \tag{5.34}$$

Now we have all the expressions necessary to calculate the derivative

$$D\mathcal{S}(\mathcal{A}, b, c) = D(\phi \circ S \circ Q(\mathcal{A}, b, c)) = D\phi(z) DS(Q) DQ(\mathcal{A}, b, c). \tag{5.35}$$

The function $Q(\mathcal{A},b,c)$ is differentiable with derivative

$$DQ(\mathcal{A},b,c) = \begin{pmatrix} 0 & D\mathcal{A}^T & Dc \\ -D\mathcal{A} & 0 & Db \\ -Dc^T & -Db^T & 0 \end{pmatrix}. \tag{5.36}$$

The derivative of $\phi(z)$ is defined in (5.31).

One can equivalently formulate the reverse or adjoint mode expressions for Automatic Differentiation

$$D\mathcal{S}^T(\mathcal{A},b,c) = DQ^T(\mathcal{A},b,c)D\mathcal{S}^T(Q)D\phi^T(z). \tag{5.37}$$

[Griewank and Walther, 2008] give the additional interpretation that the expression in (5.35) describes how sensitive the solution (x,y,s) is to small perturbations $(d\mathcal{A},db,dc)$

$$(dx,dy,ds) = D\mathcal{S}(\mathcal{A},b,c)(d\mathcal{A},db,dc) \tag{5.38}$$

and [Agrawal et al., 2020] state that the adjoint $D\mathcal{S}^T$ describes how sensitive the inputs $d\mathcal{A}$, db and dc are to a slight perturbation of the solution (x,y,s)

$$(d\mathcal{A},db,dc) = D\mathcal{S}^T(\mathcal{A},b,c)(dx,dy,ds) = D^TQ(\mathcal{A},b,c)D^TS(z)D^T\phi(z)(dx,dy,ds). \tag{5.39}$$

This is the basis of our backpropagation. For a perturbation (dx,dy,ds) we find

$$(d\mathcal{A},db,dc) = -D^TQ(\mathcal{A},b,c)DN_Q(z,Q)^TR((Q-I)D\Pi_{\mathbf{K}}(z)+I)^{-T}D\phi(z)^T(dx,dy,ds) \tag{5.40}$$

Expanding one expression at a time, we can define $g = -((Q-I)D\Pi(z) + I)^{-1}(dx,dy,ds)+$ and lastly calculate

$$(d\mathcal{A},db,dc) = D^TQ(\mathcal{A},b,c)DN_Q(z,Q)^Tg = D^TQ(\mathcal{A},b,c)g\Pi_{\mathbf{K}}(z)^T, \tag{5.41}$$

where the rearrangement of the terms comes from the derivative N_Q ((5.26)). This concludes the sections on the derivation of backpropagation through conic programs.

5.7 Differentiating the conic projection

As shown in the previous section, the differentiation of conic programs ultimately revolves around the projection on cones as well as the derivative of those projections. For a general cone, these projections and gradients could be difficult to compute. In the examples to follow we need to deal with Semidefinite Programs, a subset of conic programs limited to the cone of (symmetric) positive semidefinite matrices, $\mathbb{S}^n$ along with the zero cone, $0 \in \mathbb{R}^n$ and the non-negative cone, $\mathbb{R}^n_+$.

For the purposes of this thesis, the interesting cones are therefore the zero cone, $\{0\}$, the non-negative cone, $\mathbb{R}^n_+$, the real cone $\mathbb{R}$, and the cone of symmetric positive definite matrices. All cones used are Cartesian products of these types of cones and for the purpose of differentiation, we need to be able to differentiate the projections.

If we let the function max, be defined elementwise, the simpler projections can be written

$$\begin{aligned}
\Pi_{\mathbb{R}^n}(z) &= z \\
\Pi_{\mathbb{R}^n_+}(z)_i &= \max(z_i, 0) \\
\Pi_0(z) &= 0 \in \mathbb{R}^n
\end{aligned} \tag{5.42}$$

with derivatives,

$$\begin{aligned}
\mathrm{D}\Pi_{\mathbb{R}^n}(z) &= I \in \mathbb{R}^{n\times n} \\
\mathrm{D}\Pi_{\mathbb{R}^n_+}(z)_{ii} &= \begin{cases} 1 & \text{if} \quad z_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \\
\mathrm{D}\Pi_0(z) &= 0 \in \mathbb{R}^{n\times n}.
\end{aligned} \tag{5.43}$$

5.8 Projection onto the set of positive semidefinite matrices

The interesting projection is the Positive semi-definite projection. A more thorough explanation is given in [Busseti et al., 2018].

The projection of a symmetric matrix onto the cone of positive semidefinite matrices can be written

$$UD^+U^T \tag{5.44}$$

where U, D are obtained from the eigenvalue decomposition of the matrix, and D^+ is the diagonal matrix with $D^+_{ii} = \max(D_{ii}, 0)$.

In [Busseti et al., 2018], the derivative of the projection onto the semidefinite cone is defined implicitly.

If we let k be the number of negative eigenvalues of X, λ^+ is a sorted vector of $\max(\lambda(X), 0)$ and $\lambda^- = -\min(\lambda(X), 0)$ then we can define the matrix B from the eigenvalues of X:

$$(B)_{ij} = \begin{cases} 0 & \text{if } i \leq k, j \leq k \\ \frac{\lambda_i^+}{\lambda_j^- + \lambda_i^+} & \text{if } i > k, j \leq k \\ \frac{\lambda_j^+}{\lambda_i^- + \lambda_j^+} & \text{if } i \leq k, j > k \\ 1 & \text{if } i > k, j > k \end{cases} \tag{5.45}$$

then for any symmetric matrices X and $\tilde{X}$, the product $D\Pi(X)(\tilde{X})$ can be written

$$D\Pi(X)(\tilde{X}) = U(B \circ (U^T \tilde{X} U))U^T, \tag{5.46}$$

where U is obtained from the diagonalization of X,

$$X = UDU^T. \tag{5.47}$$

Since the derivative (5.46) is linear, it is possible to formulate via a matrix. In the implementation, detail in section 5, we construct the matrix explicitly by calculating the projection $D\Pi(x)(e_i)$, for unit vectors $e_i \in \mathbb{R}^{n\times n}$

6

Differentiation of the PEP and line search

We have derived the semidefinite PEP problem to be investigated and have shown how to differentiate a conic optimization problem. Using those results we will now show how the PEP problem can be formulated as a conic program and how it can be differentiated. Unlike in the derivation in section 5 the matrices and vectors defining the problem, b, c and $\mathcal{A}$, are not free. There is a lot of structure required. We will instead consider them to be functions of the method parameters, θ, and instead let Q be a function of the method parameters, $Q(\theta)$. We will use the fact that the objective value of the PEP, which corresponds to the worst-case convergence rate, is embedded completely in one dual variable. Therefore we will only consider the derivative of the function mapping method parameters θ to this dual variable. For this purpose, we will find direct expressions for the gradients and need not use the perturbation formulation from [Agrawal et al., 2020], though the formulations are equivalent. The chapter is concluded with a description of some specifics concerning the implementation.

6.1 Conic reformulation of the performance estimation problem

We can now write the PEP problem on the conic form, as presented in equation P,

$$\begin{array}{rl} \min & c^T x \\ \text{subject to} & s = b - \mathcal{A}x \\ & s \in \mathcal{K}. \end{array}$$

Since the objective function is affine and the constraints all include affine functions in the variables, this is mainly a reformulation from the matrix expressions in PEP into vector expressions and convex cones. We use the identity

$$\operatorname{Tr}(EF) = \sum_{i,j} E_{ij}F_{ij} = \operatorname{vec}(E)^T \operatorname{vec}(F) \tag{6.1}$$

for symmetric matrices, E and F. The primal variables are the matrix G and the function values Δf_0 and Δf_1 and we let x represent a vectorized G matrix followed by the differences Δf_i,

$$x = \begin{pmatrix} \operatorname{vec}(G) \\ \Delta f_0 \\ \Delta f_1 \end{pmatrix} \in \mathbb{R}^{(n+3m-1)^2+2m}. \tag{6.2}$$

Since we want to rephrase the maximization problem of the PEP into a minimization problem, we change the sign of c and mention that the worst-case convergence rate will be $-c^T x$ since

$$\max_x c^T x = -\min_x -c^T x, \tag{6.3}$$

and we find the objective function

$$c = \begin{pmatrix} -\operatorname{vec}(\Sigma_1^T \Sigma_1) \\ 0 \end{pmatrix}. \tag{6.4}$$

The first constraint, $\operatorname{Tr}(\Sigma_0^T \Sigma_0 G) = 1$, can be written

$$0 = 1 - \begin{pmatrix} \operatorname{vec}(\Sigma_0^T \Sigma_0) \\ 0 \end{pmatrix}^T x \equiv 1 - \mathcal{A}_1 x \iff 1 - \mathcal{A}_1 x \in \mathcal{K}_0. \tag{6.5}$$

Each of the inequality constraints, $-e_l(\Delta f_i - \Delta f_j) + \operatorname{Tr}(E_{ij}^T M_l E_{ij} G) \geq 0$, can be represented by constraints on the form

$$\mathcal{A}_2^{ijl} x \equiv \begin{pmatrix} \operatorname{vec}(E_{ij}^T M_l E_{ij}) \\ -e_l \cdot \begin{cases} 1 & \text{if } i = 0 \\ -1 & \text{if } i = 1 \\ 0 & \text{if } i = \star \end{cases} \\ -e_l \cdot \begin{cases} -1 & \text{if } j = 0 \\ 1 & \text{if } j = 1 \\ 0 & \text{if } j = \star \end{cases} \end{pmatrix}^T x \leq 0 \quad \forall i \neq j \in \{0, 1, \star\}, \quad l \in [0, 1, \ldots, m]. \tag{6.6}$$

There are $3 \cdot 2 \cdot m$ constraints on this form and we let $\mathcal{A}_2$ denote the matrix collecting all matrices $\mathcal{A}_2^{ijl}$, for which the convexity constraints can compactly be written using the non-negative cone

$$\mathcal{A}_2 x \leq 0 \iff -\mathcal{A}_2 x \in \mathbb{R}_+^{6m}. \tag{6.7}$$

Finally, the constraint on semi-definiteness is written

$$-\mathcal{A}_3 x \equiv -\begin{pmatrix} -I & 0 \end{pmatrix} x = \operatorname{vec}(G) \in \mathbb{S}^{n+3m-1}_{vec_+}, \tag{6.8}$$

where we let $\mathbb{S}^{n+3m-1}_{vec_+}$ denote the set of vectorized symmetric positive semidefinite matrices, $\operatorname{vec}(X)$ for $X \in \mathbb{S}^{n+3m-1}_{+}$. The final conic reformulation is as follows:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t} \quad & \begin{pmatrix} 1 \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \mathcal{A}_3 \end{pmatrix} x \in \begin{pmatrix} \mathcal{K}_0 \\ \mathbb{R}^{6m}_{+} \\ \mathbb{S}^{n+3m-1}_{vec_+} \end{pmatrix}. \end{aligned} \tag{6.9}$$

For the dimensions of these problems, neither solving the optimization problem nor differentiating the SDP is prohibitively expensive.

Conic formulation of the gradient method

Let us once again examine the gradient method. Recall the expressions of $\Sigma_0^T \Sigma_0$ and $\Sigma_1^T \Sigma_1$ from chapter 4. Using these we find the objective function

$$c^T = -\begin{pmatrix} 1 & -\gamma & 0 & -\gamma & \gamma^2 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.10}$$

and the matrix

$$\mathcal{A}_1 = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix} \tag{6.11}$$

Since each row in $\mathcal{A}_2$ depends on the specific parameters β and σ, we will not write it explicitly but do note that that there are 6 rows, one for each possible ordered pair of points in $\{0, 1, \star\}$.

6.2 Differentiating the PEP

Our conic optimization problem is defined by $A(\theta)$, b and $c(\theta)$, with θ denoting the parameters of some first-order optimization method. The parameters θ will only affect the E_{ij} and Σ_i matrices. Assuming strong duality, i.e $c^T x^\star + b^T y^\star = 0$ for $x^\star$ and $y^\star$ solving the primal and dual problems, we know that the worst-case convergence rate obtained from the PEP is equal to $-c^T x^\star = b^T y^\star = y_1^\star$. This simplifies the dimensions significantly. Instead of considering the mapping

$$(A, b, c) \mapsto (x^\star, y^\star, s^\star) \tag{6.12}$$

we can consider the much simpler mapping from problem parameters, θ onto y_1, which we will name $P(\theta) : \mathbb{R}^q \longrightarrow \mathbb{R}$,

$$\mathcal{P}(\theta) : \theta \mapsto y_1^\star \tag{6.13}$$

For a first-order method, the worst-case convergence rate can now be written as a function of θ

$$\mathcal{P}(\theta) = e_{n+1}^T \phi(z) \circ S(Q) \circ Q(\theta), \tag{6.14}$$

where e_{n+1} denotes the $n+1$:th canonical basis element, and is used since $\phi(z)$ produces $(x^\star, y^\star, s^\star)$, from which only the first elemtent of $y^\star$, $y_1^\star$, is desired. The function $Q(\theta)$ is used to denote the function

$$Q(\theta) = \begin{pmatrix} 0 & \mathcal{A}(\theta)^T & c(\theta) \\ -\mathcal{A}(\theta) & 0 & b \\ -c(\theta)^T & -b^T & 0 \end{pmatrix}. \tag{6.15}$$

For a single parameter, θ_i, the derivative of $P(\theta)$ can be expressed

$$D_{\theta_i}\mathcal{P}(\theta) = e_{n+1} D\phi(z) DS(Q) D_{\theta_i} Q(\theta). \tag{6.16}$$

Since this is a scalar expression, we can transpose and find the alternative expression

$$\begin{aligned} D_{\theta_i}\mathcal{P}(\theta) = D_{\theta_i}\mathcal{P}(\theta)^T &= DQ(\theta_i)^T DS(Q)^T D\phi(z)^T e_{n+1} \\ &= \Pi_{\mathbf{K}}(z)^T DQ(\theta_i)^T (-((Q-I)D\Pi(z)+I)^{-T}) D\phi(z)^T e_{n+1}, \end{aligned} \tag{6.17}$$

with the added benefit that the relatively expensive matrix inversion, can be calculated once and reused for all derivatives, D_{θ_i}. If we store the result of the computation

$$g = -((Q-I)D\Pi(z)+I)^{-T}) D\phi(z)^T e_{n+1}, \tag{6.18}$$

and calculate the projection $\Pi_{\mathbf{K}}(z)$ beforehand, the differentiation can simply be done through the product

$$D_{\theta_i} P(\theta) = \Pi_{\mathbf{K}}(z)^T D_{\theta_i} Q^T g. \tag{6.19}$$

We find the gradient

$$D_\theta P(\theta) = \begin{pmatrix} D_{\theta_1} P(\theta) \\ D_{\theta_2} P(\theta) \\ \vdots \\ D_{\theta_q} P(\theta) \end{pmatrix} = \begin{pmatrix} \Pi_{\mathbf{K}}^T(z) D_{\theta_1} Q^T \\ \Pi_{\mathbf{K}}^T(z) D_{\theta_2} Q^T \\ \vdots \\ \Pi_{\mathbf{K}}^T(z) D_{\theta_q} Q^T \end{pmatrix} g \tag{6.20}$$

It should be noted that an equivalent expression can be gained from the reverse mode perturbation formulation in chapter 5, if choosing $dy = e_1$.

6.3 Differentiation of the Q matrix

Now all that remains is the differentiation of $A(\theta)$ and $c(\theta)$ where we note that both are formed by quadratic matrix expressions, for example,

$$E_{ij}(\theta)^T M_l E_{ij}(\theta), \tag{6.21}$$

where can be expressed coordinate-wise as

$$(C)_{pq} \equiv (E_{ij}(\theta)^T M_l E_{ij}(\theta))_{pq} = E_{ij}^{(p)T} M E_{ij}^{(q)}, \tag{6.22}$$

where $E_{ij}^{(p)}$ denotes the p:th column of E_{ij}. Using the chain rule it is possible to find the coordinate-wise derivative

$$D_{\theta_k}(C)_{pq} = (D_{\theta_k} E_{ij}^{(p)})^T M_l E_{ij}^{(q)} + (D_{\theta_k} E_{ij}^{(q)})^T M_l E_{ij}^{(p)}, \tag{6.23}$$

from which one can express the derivative

$$D_{\theta_k} C = (D_{\theta_k} E_{ij})^T M_l E_{ij} + E_{ij}^T M_l (D_{\theta_k} E_{ij}), \tag{6.24}$$

which gives the following derivatives

$$\begin{cases} D_{\theta_k} c = \begin{pmatrix} -\operatorname{vec}\left(D_{\theta_k} \Sigma_1(\theta)^T \Sigma_1(\theta_k)\right) \\ \mathbf{0} \end{pmatrix} \\ D_{\theta_k} \mathcal{A} = \begin{pmatrix} \mathbf{0} \\ \operatorname{vec}\left(D_{\theta_k} E_{ij}^T M_l E_{ij}\right)^T \forall i,j,l \\ \mathbf{0} \end{pmatrix} \\ D_{\theta_k} b = \mathbf{0} \end{cases} \tag{6.25}$$

This is used to construct the matrix

$$D_{\theta_l} Q(\theta) = \begin{pmatrix} 0 & D_{\theta_k}\mathcal{A}^T & D_{\theta_k} c \\ -D_{\theta_k}\mathcal{A} & 0 & 0 \\ -D_{\theta_k} c^T & 0 & 0 \end{pmatrix} \tag{6.26}$$

Note that all these expressions can be formulated in terms of the matrices A, B, C and D and their derivatives.

$$\begin{aligned} D_{\theta_k} \Sigma_1^T \Sigma_1 = &\begin{pmatrix} D_{\theta_k} A & D_{\theta_k} B & 0 & (D_{\theta_k} B) N \end{pmatrix}^T \begin{pmatrix} A & B & 0 & -BN \end{pmatrix} \\ &+ \begin{pmatrix} A & B & 0 & -BN \end{pmatrix}^T \begin{pmatrix} D_{\theta_k} A & D_{\theta_k} B & 0 & (D_{\theta_k} B) N \end{pmatrix} \end{aligned} \tag{6.27}$$

, where similar expressions can be obtained for the row-wise derivative of the $\mathcal{A}$ matrix:

$$\begin{cases} D_{\theta_k}(E_{01}^T M_l E_{01}) = (D_{\theta_k} E_{01})^T M_l E_{01} + E_{01}^T M_l D_{\theta_k} E_{01} (D_{\theta_k} E_{01}) \\ \text{where } D_{\theta_i} E_{01} = \begin{pmatrix} (D_{\theta_k} C)(A - I) + C D_{\theta_k} A & (D_{\theta_k} C) B + C (D_{\theta_k} B) - D_{\theta_k} D & 0 & D_{\theta_k} D \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{cases}, \tag{6.28}$$

with similar expressions for the other $D_{\theta_i} E_{ij}$

Differentiation for the gradient method

Let us once again show this for the gradient method. We can consider the derivative $D_\theta c$, where we recall that $c = \Sigma_1^T \Sigma_1$ for

$$\Sigma_1 = \begin{pmatrix} 1 \\ -\gamma \\ 0 \end{pmatrix}. \tag{6.29}$$

There is only one parameter, γ and the derivative is

$$\begin{aligned}
-D_\gamma(\Sigma_1^T \Sigma_1) &= -(D_\gamma \Sigma_1)^T \Sigma_1 - \Sigma_1^T (D_\gamma \Sigma_1) \\
&= -\begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & -\gamma & 0 \end{pmatrix} - \begin{pmatrix} 1 \\ -\gamma \\ 0 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & 1 & 0 \\ 1 & -2\gamma & 0 \\ 0 & 0 & 0 \end{pmatrix}
\end{aligned} \tag{6.30}$$

As a result

$$D_\gamma c = \begin{pmatrix} \mathrm{vec}(-D_\gamma(\Sigma_1^T \Sigma_1)) \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 & -2\gamma & 0 & \dots & 0 \end{pmatrix}^T \tag{6.31}$$

One can differentiate the $\mathcal{A}$ matrix similarly.

6.4 Pseudocode for differentiating the PEP

For each iteration, we need to find solutions to the PEP-problem, (x, y, s). From here, a solution to the self-dual embedding can be constructed

$$z = (x, y - s, 1). \tag{6.32}$$

Note that the dual problem to (6.9) (as defined in (D)) has the objective function

$$b^T y = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix} y = y_1. \tag{6.33}$$

From the expressions detailed in (6.17), we initially calculate,

$$dz = D\phi(z)^T e_{n+1} = \begin{pmatrix} I & 0 & 0 \\ 0 & D\Pi_{\mathcal{K}^*}(z_2)^T & D\Pi_{\mathcal{K}^*}(z_2)^T - I \\ -x & -y & -s \end{pmatrix} e_{n+1} = \begin{pmatrix} 0 \\ (D\Pi_{\mathcal{K}^*}(y-s))^T e_1 \\ -y_1 \end{pmatrix} \tag{6.34}$$

and ultimately

$$g = ((Q - I) D\Pi_{\mathcal{K}}(z) + I)^{-T} dz. \tag{6.35}$$

Lastly, we can calculate the projection onto $\mathbf{K}$ and the matrix for the derivative $D\Pi_{\mathbf{K}}$. Given a method, an objective function and constants for strong convexity and smoothness, σ and β, we can construct a conic problem, which can be differentiated using algorithm 1. In order to deal with ill-conditioned problems, we will use a least-squares solver to replace the matrix inverse in (5.34).

Algorithm 1 Differentiating the PEP

```
procedure DIFFERENTIATEPEP($\theta$)
    $\mathcal{A}, b, c \leftarrow \mathcal{A}(\theta), e_1 \in \mathbb{R}^m, c(\theta)$
    $Q \leftarrow \begin{pmatrix} 0 & \mathcal{A}^T & c \\ -\mathcal{A} & 0 & b \\ -c^T & -b^T & 0 \end{pmatrix}$
    $x, y \leftarrow \text{solve}(\theta), \text{solveDual}(\theta)$
    $s \leftarrow b - \mathcal{A}x$
    $z \leftarrow (x, y - s, 1)$
    $dz \leftarrow \begin{pmatrix} 0 \\ D\Pi_{\mathcal{K}^*}(y-s)^T e_1 \\ -y^T e_1 \end{pmatrix}$
    $g \leftarrow \text{leastSquares}(((Q-I)D\Pi_{\mathbf{K}}(z)+I)^T, -dz)$
    $\Pi_{\mathbf{K}} \leftarrow \text{project}(z, \mathbf{K})$
    for $i = 0$ to length($\theta$) do
        $DQ \leftarrow \begin{pmatrix} 0 & D_{\theta_i}\mathcal{A}(\theta)^T & D_{\theta_i}c(\theta) \\ -D_{\theta_i}\mathcal{A}(\theta) & 0 & 0 \\ -D_{\theta_i}c(\theta) & 0 & 0 \end{pmatrix}$
        $\nabla_\theta[i] \leftarrow \Pi_{\mathbf{K}}^T DQ^T g$
    end for
    return $\nabla_\theta$

end procedure
```

The costly operations are mainly the primal and dual solves of the PEP, followed by the least-squares solver. The differentiation from [Agrawal et al., 2020], use the iterative method LSQR which only requires matrix-vector multiplications, for solving the least squares problem. The numerical inaccuracy of the iterative solver manifested itself in some of test in the following chapter, resulting in early termination and worse results. Instead, the built-in least squares solver, based on the SVD, is used without any notable performance penalty. For the problem at hand, the expensive operation is the SDP-solve. Some tests were made using the SCS-solver used in the implementation by [Agrawal et al., 2020], but the accuracy was significantly worse than with a case-specific SDP-solver. Our choice turned to the MOSEK SDP-solver. For these examples, the optimization problem is small enough for expensive but more exact solvers to be worthwhile.

7

Implementation and Results

The previous chapter describes how to differentiate the Performance Estimation Problem. Now we turn to implementation and results. In this chapter, we give some results, using the differentiation derived in chapter 6. Initially, we motivate the choice of line search. This is followed by the results for a few methods.
The main examples we investigate are the Douglas-Rachford method, where there are analytical, tight rates for the worst-case convergence, and the Davis-Yin operator splitting method, where no such exact bounds are known. For verification, some code was borrowed from the PEP implementation from [Upadhyaya et al., 2023].

7.1 Implementation of gradient-based optimization

The problem is as follows:

$$\min_{\theta} \mathcal{P}(\theta) \tag{7.1}$$

for $\mathcal{P}(\theta)$ defined as the solution to the PEP as a function of the method parameters θ, for a given first-order method and strong convexity and Lipschitz constants. Our approach uses the gradients computed as described in algorithm 1. In addition, some kind of line search or step size choice is needed. Some different choices were tried: constant step sizes, fixed, decreasing step sizes and inexact, backtracking, line-search based on the Armijo rule. Constant step sizes,

$$\theta_{k+1} = \theta_k - d \cdot \nabla_\theta \mathcal{P}(\theta), \tag{7.2}$$

do not always converge, even for the simplest examples. This is due to non-differentiability in the minimum, for $\mathcal{P}(\theta)$. A simple example explaining this behavior is the following: Let $f(x) = |x|$, with gradient $\nabla_x f = \mathrm{sign}(x)$ and assume that $x_k \in (0, d)$. Then, when using the gradient method with fixed step sizes, one enters the cycle

$$\begin{aligned} x_{k+1} &= x_k - d \cdot \mathrm{sign}(x_k) = x_k - d < 0 \\ x_{k+2} &= x_{k+1} - d \cdot \mathrm{sign}(x_{k+1}) = x_{k+1} + d = x_k. \end{aligned} \tag{7.3}$$

A simple solution would be decreasing step sizes:

$$\theta_{k+1} = \theta_k - d_k \nabla_\theta \mathcal{P}(\theta), \tag{7.4}$$

where $d_k = \frac{d_1}{k+1}$ is one possible choice. Due to slow convergence, this was instead replaced with in-exact backtracking line search, using the Armijo rule (see algorithm 2). The main idea behind in-exact line search is to only find a satisfactory step size, for which the function value decreases enough according to some criterion, instead of attempting to find the optimal step size, which would be a lot more expensive. For multi-dimensional problems, like the ones considered here, this is often a better approach than more exact line searches, which require more function evaluations. An introduction to inexact line search and the Armijo rule can be found in [Böiers, 2010].

Algorithm 2 Inexact backtracking line search

```
procedure LINESEARCH(method, $\theta, \nabla_\theta, \rho_0$)
    $d \leftarrow 1$
    $c \leftarrow \frac{1}{2}$
    $m \leftarrow \|\nabla_\theta\|_2^2$
    while $\lambda \cdot m \leq$ TOL do
        $\rho_1 \leftarrow$ solveDual(method, $\theta + d \cdot \nabla_\theta$)[0]
        $\rho_2 \leftarrow$ solveDual(method, $\theta + 2 \cdot d \cdot \nabla_\theta$)[0]
        if $\rho_1$ is None or $\rho_0 \leq \rho_1 + m \cdot c \cdot d$ then
            $d \leftarrow d/2$
        else if $\rho_2$ is not None and $\rho_0 \geq \rho_2 + 2 \cdot m \cdot c \cdot d$ then
            $d \leftarrow 2 \cdot d$
        else
            break
        end if
    end while
    return $(\theta + d \cdot \nabla_\theta, \rho_1)$
end procedure
```

In all examples, we seek for a linear convergence rate, ρ^2, for the function $\sum_{i=1}^{n} \|x^{(i)} - x_*^{(i)}\|_2^2 = \|x - x_*\|_2^2$.

7.2 Numerical results

As an initial example, we can investigate the Douglas–Rachford operator-splitting method, as explained in (DR), for different classes of functions f_1 and f_2. There are known, tight convergence-rate bounds, that can be compared to. The initial example seeks to demonstrate convergence, at some distance from the solution. Starting at

Algorithm 3 Optimization over the method parameters

```
procedure MINIMIZEPARAMS($\theta$, method)
    for $i = 1$ to MAXITER do
        $\nabla_\theta, x, y \leftarrow$ differentiatePEP(method, $\theta$)
        $\rho_0 \leftarrow y[0]$
        $\theta, \rho_1 \leftarrow$ lineSearch(method, $\theta, \nabla_\theta, \rho_0$)
        if $\rho_0 - \rho_1 \leq$ TOL then
            break
        end if
        $\rho_0 \leftarrow \rho_1$
    end for
end procedure
return $(\theta, \rho_1)$
```

half of the optimal parameter values, the proposed method converges in most cases Fot the last example of the Douglas–Rachford method, a better bound on the convergence rate is found when using the adaptive Douglas–Rachford method, which has the structure

$$\begin{aligned} x_{k+1} &= x_k - \lambda \begin{pmatrix} \gamma_2 & \gamma_2 \end{pmatrix} u_k \\ y_k &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} x_k + \begin{pmatrix} -\gamma_1 & 0 \\ -\gamma_1 - \gamma_2 & -\gamma_2 \end{pmatrix} u_k \end{aligned} \tag{7.5}$$

Lastly, the Davis–Yin method is investigated, and compared to the adaptive Davis–Yin method. A small comparison is made between two methods using theoretical gradients as computed in algorithm 1, and one using a finite difference approximation.

Douglas-Rachford

The Douglas–Rachford operator splitting method is used to minimize the sum of two functions, f_1 and f_2. For the first example, consider the case where $f_1 \in \mathcal{F}_{\sigma,\beta}$ and $f_2 \in \mathcal{F}_{0,\infty}$. In [Giselsson, 2015], it is shown that the theoretically tight, worst-case convergence rate for the Douglas–Rachford method is

$$\rho = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}, \tag{7.6}$$

in this case. The condition number, κ, is defined as $\kappa = \frac{\beta}{\sigma}$. We will instead investigate the scaled problem, $f_1 \in \mathcal{F}_{\frac{1}{\sqrt{\kappa}}, \sqrt{\kappa}}$. This is the problem obtained, when dividing the functions f_1 and f_2 by the constant $\sigma\sqrt{\kappa} = \sqrt{\beta}\sqrt{\sigma}$. There are advantages to this choice. Firstly, it simplifies the comparison since the scaled problem always has optimal Douglas–Rachford parameters, $\gamma = 1$ and $\lambda = 2$, but more importantly, there seem to be some numerical benefits as the implemented solver seems to perform better, especially for large condition numbers. This scaling has no impact on the condition number, κ, so the optimal rate is the same. For the first example, we investigate how well this implementation converges. The initial Douglas–Rachford parameters are chosen to be half of the optimal ones. Starting at $\gamma = \frac{1}{2}$ and $\lambda = 1$, we obtain the optimal parameters in most cases. The results are significantly worse for other choices of scaling of f_1 and f_2. A hypothesis is that this is due to the M_l matrix (4.17), and its products. The condition number of the matrix M_l is much lower for this choice of scaling, although that is probably not the full explanation.

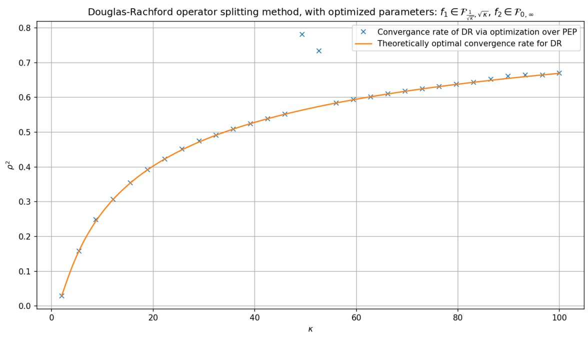

Figure 7.1 Obtained, optimal convergence rate, starting with initial parameters $\gamma = \frac{1}{2}$ and $\lambda = 1$

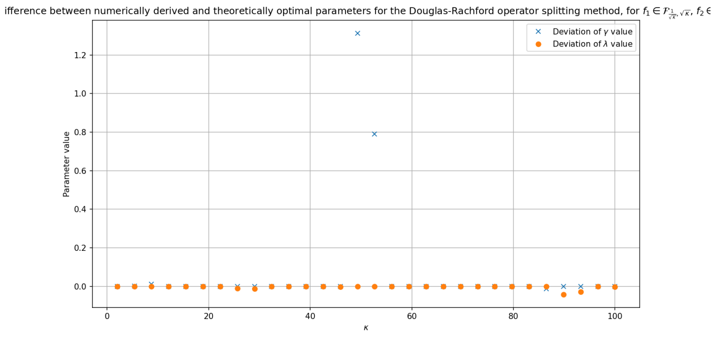

Figure 7.2 Resulting optimal parameters, starting with initial parameters $\gamma = \frac{1}{2}$ and $\lambda = 1$

The resulting optimal linear convergence rate and parameters correspond to the theoretical optima (see figure 7.1 and 7.2), except in a few cases.

For the next example, the functions are assumed to be $f_1 \in \mathcal{F}_{0,\sqrt{\kappa}}$ and $f_2 \in \mathcal{F}_{\frac{1}{\sqrt{\kappa}},\infty}$. Solving for increasing values of $\kappa \in [1, 2, \dots, 40]$, using the previous optimal parameters, gives good results (see figure 7.3). Interestingly, the adaptive Douglas–Rachford method performs better for this example. The optimal parameters for the Douglas–Rachford method and the adaptive Douglas–Rachford method can be seen in figures 7.5 and 7.4 respectively.

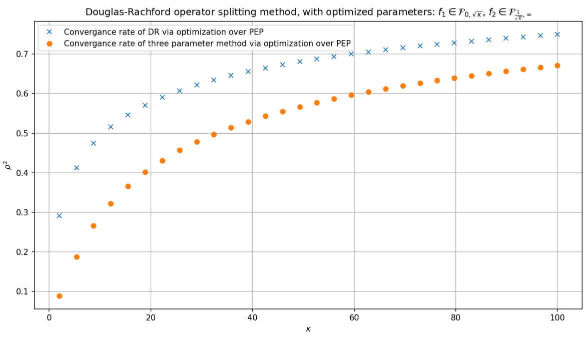

Figure 7.3 Obtained optimal squared convergence rate for the classical (blue) and the adaptive (orange) Douglas-Rachford methods.

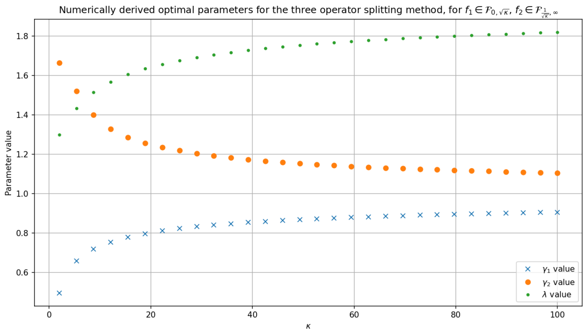

Figure 7.4 Optimal parameters for the adaptive Douglas–Rachford method.

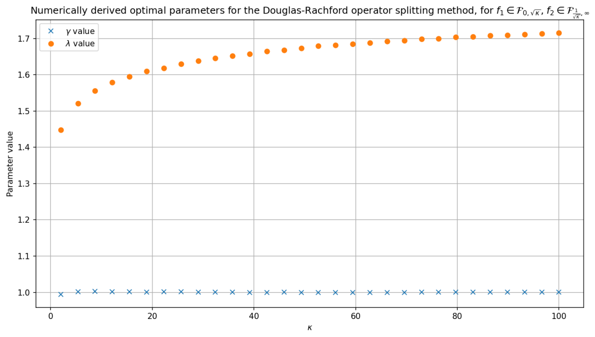

Figure 7.5 Optimal parameters for the Douglas–Rachford method

The Davis–Yin three operator splitting method

For the last example, the Davis–Yin method is investigated, for the case of functions, $f_1 \in \mathcal{F}_{0,\beta}$, $f_2 \in \mathcal{F}_{1,2}$ and $f_3 \in \mathcal{F}_{0,\infty}$. Recall the structure of the Davis–Yin method

$$\begin{aligned} x_{k+1} &= x_k - \lambda \begin{pmatrix} \gamma & \gamma & \gamma \end{pmatrix} u_k \\ y_k &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} x_k + \begin{pmatrix} -\gamma & 0 & 0 \\ -\gamma & 0 & 0 \\ -2\gamma & -\gamma & -\gamma \end{pmatrix} u_k \\ u_k &\in \partial \mathbf{f}(y_k). \end{aligned} \tag{7.7}$$

Similarly to before, the worst-case convergence rate should be invariant to scaling of the three functions, with the same constant. Therefore, we consider the case $f_1 \in \mathcal{F}_{0,\sqrt{\beta}}$, $f_2 \in \mathcal{F}_{\frac{1}{\sqrt{\beta}},\frac{2}{\sqrt{\beta}}}$, $f_3 \in \mathcal{F}_{0,\infty}$ and rescale the parameters, for the original problem afterward. For the new problem, the gradients and subdifferentials will simply decrease by a factor $1/\sqrt{\beta}$. We can simply scale γ with the factor $\sqrt{\beta}$ to counteract this an ensure the same x_k and y_k iterates.

The scaled parameter values were always initiated to the previously obtained optimal parameters. For $\beta = 1$, the initial parameters were $\lambda = 1$, $\gamma = \frac{1}{2}$.

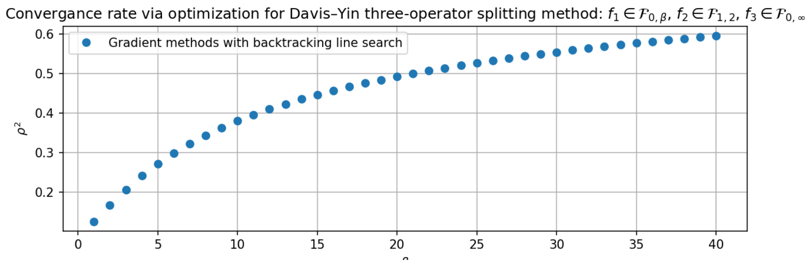

Figure 7.6 Optimal convergence rate for the Davis-Yin method.

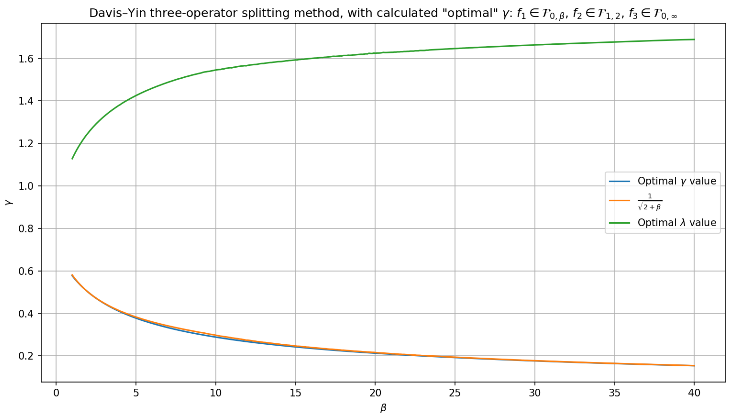

Figure 7.7 Optimal parameters for the Davis-Yin method.

These rates in figure 7.7, improvements, compared to the theoretically stable parameter choices of $\lambda = 1$, $\gamma = \frac{1}{2}$, as tested in [Upadhyaya et al., 2023]. Interestingly, the optimal γ value is very close to $\frac{1}{2+\beta}$.

A simple comparison of accuracy

The inexact line-search with gradients from algorithm 1 is compared to two alternative solvers: the default SCIPY minimization algorithm, using BFGS, and BFGS using the gradients from algorithm 1. Comparing the same example of the Davis–Yin method as previously but with uniformly random initial parameters $\gamma, \lambda \in [0,1]$ and $\beta = 1, 2, \ldots, 40$, the inexact line search is both faster and more accurate than the SCIPY-implementation of BFGS. If using the gradients from algorithm 1 with BFGS, the result is more accurate, but the time increases significantly.

Method	Computation time (s)	Average worst-convergence rate (ρ^2)
Armijo (grad from alg. 1)	207	0.4570
BFGS (default)	212	0.5333
BFGS (grad from alg. 1)	290	0.4568

Table 7.1 Total computation time and average objective value, for 40 solves with $f_1 \in \mathcal{F}_{0,\sqrt{\beta}}$, $f_2 \in \mathcal{F}_{1/\sqrt{\beta},2/\sqrt{\beta}}$ and $f_3 \in /,\infty$. The BFGS method is the default solver of scipy.minimize, which uses BFGS with finite difference approximations for gradients. The second BFGS method uses the gradients from algorithm 1.

An adaptive Davis–Yin method

Just like for Douglas-Rachford, it is possible to have different parameters, γ_1 and γ_2 for the two proximal evaluations, while using the Davis-Yin method. The convergence of this adaptive Davis-Yin method can be seen in figure 7.8. The convergence rate is better than for classical Davis-Yin.

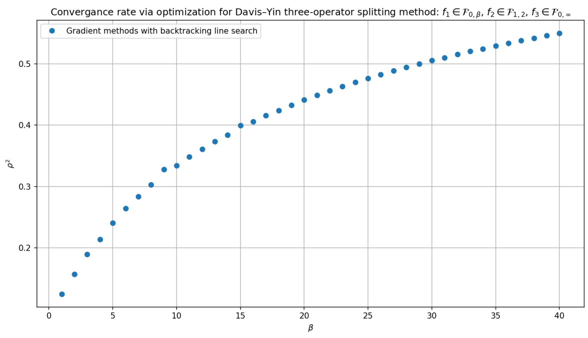

Figure 7.8 Obtained optimal squared convergence rate for an adaptive three-parameter Davis-Yin method.

8 Conclusions

We have investigated the problem of finding optimal parameters of first-order optimization methods, using an adaptation of a recent method for differentiating convex optimization problems. The problem of finding optimal parameters is generally non-convex and it is clear from tests, that for bad initial parameter guesses or less well-behaved examples, it is not guaranteed that the method finds the optimal parameters. For the cases investigated, though, the results are accurate, and we find the theoretically optimal parameters of the Douglas-Rachford method, as well as good results in cases where there are no theoretical optima (to my knowledge), like the adaptive Douglas–Rachford method or the classical or the adaptive Davis–Yin methods.

The main benefit of the theoretical gradients, as calculated in algorithm 1, is a reduced need for the comparatively expensive solves of the semidefinite programs, compared to minimization without gradients, and better accuracy than other methods for finding gradients, like finite differences. The differentiation revolves around the solutions of the primal and dual problems, as well as a least squares problem, all of which can be solved accurately and efficiently. This method performs well, both with in-exact line search and with BFGS. The method seems sensitive to numerical errors in the SDP or least squares solves, based on some tests. Replacing more exact solvers, i.e a case-specific SDP solver and backward stable least-squares solvers, leads to significantly worsened results, for the examples in this report.

We also show a result on the possible parametrizations of first-order methods, in the state space context and the case of lifting 1, however it is possible to generalize this approach and find parameterizations also for lifting larger than 1. An interesting experiment would be to investigate these more general parameterizations. It is in these kinds of higher dimensional problems, where this form of gradient-based optimization of parameters would be expected to outperform shine, compared to the gradient-free method used by [Ryu et al., 2020] or methods using the central-difference approximations of the gradients, which would be significantly more expensive. It could also be expected that these higher-dimensional problems with more parameters are less well-behaved.

The proposed method for optimizing parameters is general and can be used for any state space representable algorithm, though there are no guarantees of convergence to the optimal parameters. Some other optimization algorithms were tested with mixed results, but not included in the report.

Bibliography

Agrawal, A., S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi (2020). *Differentiating through a cone program*. arXiv: `1904.09043 [math.OC]`.

Böiers, L.-C. (2010). *Mathematical methods of optimization*. Studentlitteratur.

Boyd, S. P. and L. Vandenberghe (2011). *Convex optimization*. Cambridge Univ. Pr.

Busseti, E., W. Moursi, and S. Boyd (2018). *Solution refinement at regular points of conic problems*. arXiv: `1811.02157 [math.OC]`.

Dao, M. N. and H. M. Phan (2019). "Adaptive douglas–rachford splitting algorithm for the sum of two operators". *SIAM Journal on Optimization* **29**:4, pp. 2697–2724. DOI: `10.1137/18m121160x`. URL: `https://doi.org/10.1137%2F18m121160x`.

Dontchev, A. L. and R. T. Rockafellar (2009). "1b. the classical implicit function theorem". In: *Implicit functions and solution mappings*. Springer.

Drori, Y. and M. Teboulle (2013). "Performance of first-order methods for smooth convex minimization: a novel approach". *Mathematical Programming* **145**:1–2, pp. 451–482. DOI: `10.1007/s10107-013-0653-0`.

Giselsson, P. (2015). "Tight linear convergence rate bounds for douglas-rachford splitting and admm". *2015 54th IEEE Conference on Decision and Control (CDC)*. DOI: `10.1109/cdc.2015.7402716`.

Griewank, A. and A. Walther (2008). *Evaluating Derivatives*. Second. Society for Industrial and Applied Mathematics. DOI: `10.1137/1.9780898717761`. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9780898717761`. URL: `https://epubs.siam.org/doi/abs/10.1137/1.9780898717761`.

Lessard, L., B. Recht, and A. Packard (2016). "Analysis and design of optimization algorithms via integral quadratic constraints". *SIAM Journal on Optimization* **26**:1, pp. 57–95. DOI: `10.1137/15M1009597`. eprint: `https://doi.org/10.1137/15M1009597`. URL: `https://doi.org/10.1137/15M1009597`.

Parikh, N. (2014). *Proximal algorithms*. DOI: `10.1561/9781601987174`.

Ryu, E. K., A. B. Taylor, C. Bergeling, and P. Giselsson (2020). “Operator splitting performance estimation: tight contraction factors and optimal parameter selection”. *SIAM Journal on Optimization* **30**:3, pp. 2251–2271. DOI: `10.1137/19M1304854`. eprint: `https://doi.org/10.1137/19M1304854`. URL: `https://doi.org/10.1137/19M1304854`.

Taylor, A. B., J. M. Hendrickx, and F. Glineur (2016). “Smooth strongly convex interpolation and exact worst-case performance of first-order methods”. *Mathematical Programming* **161**:1–2, pp. 307–345. DOI: `10.1007/s10107-016-1009-3`.

Upadhyaya, M., S. Banert, A. B. Taylor, and P. Giselsson (2023). *Automated tight lyapunov analysis for first-order methods*. arXiv: `2302.06713 [math.OC]`.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* June 2023 *Document Number* TFRT-6204
Author(s) Anton Åkerman	*Supervisor* Pontus Giselsson, Dept. of Automatic Control, Lund University, Sweden Yiannis Karayiannidis, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Optimizing First-Order Method Parameters via Differentiation of the Performance Estimation Problem

Abstract

This thesis treats the problem of finding optimal parameters for first-order optimization methods. In part, we use the Performance Estimation Problem (PEP), a framework for convergence analysis of first-order optimization methods. The fundamental idea of the PEP is to formulate the problem of finding the worst-case convergence rate of a first-order optimization algorithm, as an optimization problem. We also use recent methods for differentiating convex optimization problems. The goal is to explore the use of gradient-based methods for finding optimal parameters of first-order optimization methods, within the context of the Performance Estimation Problem. By differentiating the PEP, we can find gradients which can be used in an attempt to search for optimal method parameters.
We consider the state space representation of first-order methods, which include many well-known first-order operator splitting methods. We propose a gradientbased algorithm for optimizing first-order method parameters, based on the differentiation algorithm from [Agrawal et al., 2020] and the PEP representations from [Upadhyaya et al., 2023], and show decent results. This is a heuristic approach to a non-convex optimization problem, but it works well for the Douglas–Rachford and Davis–Yin operator splitting methods. The results seem to agree with the theoretically optimal parameters for the Douglas–Rachford method, and the obtained convergence rates for the Davis–Yin method are better than the ones found in [Upadhyaya et al., 2023], using fixed parameters. The presented results, concern only those two methods, but the proposed algorithm is general. Based on some limited testing, this problem seems sensitive to numerical inaccuracy, and as a consequence, our approach using more exact gradients seems to outperform the built-in solver from SCIPY, which uses approximate gradients, terminating faster with comparable (or better) accuracy.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-64	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/