

Contributions to Performance Estimation Problems for the Analysis and Design of First-Order Optimization Methods

Yassine Kamri

Thesis submitted in partial fulfillment
of the requirements for the degree of
Docteur en sciences de l'ingénieur

Dissertation committee:

Prof. François Glineur (UCLouvain, *Advisor*)
Prof. Julien M. Hendrickx (UCLouvain, *Advisor*)
Prof. Panagiotis Patrinos (KULeuven, *Reader*)
Prof. Ion Necoara (UPB, *Reader*)
PhD. Damien Scieur (SAIL Montréal, *Reader*)
Prof. Geovani Grapiglia (UCLouvain, *Secretary*)
Prof. Jean-Charles Delvenne (UCLouvain, *Chair*)

Version of July 4, 2025.

Contents

1	Introduction	5
1.1	Mathematical Optimization	5
1.2	Large-scale Optimization and First-Order Methods	6
1.3	Worst-case Analysis of First-Order Methods	7
1.4	Performance Estimation Problem (PEP)	8
1.5	Objectives of the thesis	12
1.6	Outline of the Thesis and List of Publications	13
2	Background: Performance Estimation Problems	15
2.1	Introduction and Problem Statement	15
2.2	Smooth (Strongly)-Convex Functions	18
2.3	Convex Formulation of the Performance Estimation Problem	20
2.4	Use Case Example of PEP: Worst-case Analysis of Gradient Descent for L -Smooth Convex Functions	26
2.5	Design of Optimal First-Order Algorithms for the Functional class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$	28
2.6	Conclusion	34
3	Performance Estimation of First-Order Block Coordinate-Wise Algorithms	35
3.1	Introduction	35
3.2	Class of Coordinate-wise Smooth Functions	42
3.3	Convex PEP for Fixed-Step BCD Algorithms on the Functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$	49
3.4	Numerical Experiments on the Convergence of Block Coordinate-wise Algorithms on the class $\mathcal{F}_{0,L}(\mathbb{R}^d)$	56

3.5	A convex Performance Estimation Problem framework for the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$	67
3.6	Worst-case Behavior of Cyclic Coordinate Descent	78
3.7	Numerical Experiments on the Convergence of Block Coordinate-wise Algorithms on the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$	88
3.8	Conclusion	102
4	Performance Estimation of First-Order Algorithms over Hölder-Smooth Convex Functions	103
4.1	Introduction	103
4.2	Hölder-Smooth Convex Functions	107
4.3	Convex Performance Estimation Problem	111
4.4	Worst-case Analysis of Fixed-Step First-Order Algorithms for the Functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$	116
4.5	Conclusion	123
5	Numerical Design of Optimized First-Order Algorithms	125
5.1	Introduction	125
5.2	Problem Formulation	127
5.3	Numerical Design Methods for First-Order Optimization Algorithms	132
5.4	Comparison on the benchmark of Optimizing Memoryless Gradient Descent	139
5.5	Numerical Design of First-Order Methods using SLM	146
5.6	Conclusion	175
6	Conclusion	177
6.1	Summary of the Contributions	177
6.2	Future Perspectives	179

Abstract

Performance Estimation Problems (PEPs) deal with the worst-case convergence analysis of fixed-step first-order optimization algorithms. This thesis extends PEPs to analyze different classical convex optimization settings, for which they can be framed and solved as convex optimization problems.

First, we construct a convex PEP formulation for block coordinate-wise algorithms over the class of coordinate-wise smooth convex functions. We use this tool to analyze several popular algorithms such as cyclic coordinate descent and alternating minimization, and provide tighter numerical convergence rates compared to those available in the literature. We also give new insights about the behavior of these algorithms, notably by characterizing the dependence of the worst-case of cyclic coordinate descent on the number of blocks, estimating numerically optimal constant step sizes for cyclic coordinate descent, and analyzing the efficiency of deterministic variants of random accelerated coordinate descent algorithms.

Secondly, we extend the PEP framework for the analysis of gradient descent over Hölder-smooth convex functions, propose a fixed step-size strategy in this context, demonstrate the convergence of the algorithm and compare its convergence rate to the one of universal gradient methods.

Finally, we extend the PEP framework for the automatic design of optimized fixed-step first-order algorithms. We propose several methods for the numerical design of fixed-step first-order algorithms, and test them in different convex optimization settings, including those previously considered in this thesis. This leads to numerically optimized algorithms that are frequently displaying faster convergence rates compared to the classical algorithms.

Acknowledgments

A PhD thesis is a long journey, and one that cannot be completed without the support and guidance of many along the way. I have been fortunate to receive unwavering support from many people, for which I am truly grateful.

First and foremost, I would like to thank my two advisors, Professors Julien Hendrickx and François Glineur, for giving me the opportunity to pursue a PhD in the Mathematical Engineering Department, and for their invaluable guidance, time, and support throughout this journey. I would also like to express my gratitude to Prof. Jean-Charles Delvenne, Prof. Panagiotis Patrinos, Prof. Geovani Grapiglia, Prof. Ion Necoara and Dr. Damien Scieur for kindly agreeing to serve on the jury of this thesis, for the time they dedicated to reviewing my work, and for the valuable feedback they provided to help improve this manuscript.

I would also like to express my thanks to all the colleagues and friends I had the pleasure of meeting at the Euler Building, whose support has been instrumental to the completion of this thesis. In particular, I would like to thank Ousmane, Sofiane, and Erwan, with whom I shared an office—thank you for making office A.021 such a pleasant and enjoyable place to work. I am also grateful to the members of my research group—Anne, Seb, Charles, Antoine, Nizar, and Martina—for the many moments we shared, both at the office and beyond. Thank you for the great team spirit we created, and for the kindness and mutual support we showed one another. I would also like to thank Virginie, Julien, Adrien, Philémon, Rémi, Matteo, Lucas, Renato, Clémence, Astrid D., Guillaume B., Jana and John. I was very fortunate to have you as friends during the challenging times of my PhD. I would like to thank the administrative and technical staff—Marie-Christine, Pascale,

Astrid L., François W. and Etienne—for their kindness and the outstanding work they do in keeping the department running as smoothly as possible.

Finally, and most importantly, I would like to express my love and gratitude to my family: my parents, Laila and Abdelkrim, and my sister, Hiba. To my parents—thank you for your unconditional love, unwavering support, and every effort and personal sacrifice you made for me and my education. None of this would have been possible without you. To my sister, my best friend—thank you for always being by my side and for always believing in me.

1

Introduction

1.1 Mathematical Optimization

Mathematical optimization can be informally defined as the systematic process of finding the best solution, to a problem from a set of feasible alternatives, X , guided by a well-defined objective function f , i.e., finding a solution, x , to the problem:

$$\min_{x \in X} f(x).$$

Optimization lies at the heart of numerous scientific, engineering, and decision-making problems. Indeed, many practical problems can be framed within the formalism of the above optimization problem. For instance, we refer the reader to [82, 92, 77] for examples in automatic control, machine learning, and signal processing. Although almost all practical engineering or scientific problems can be formulated within the optimization framework, the resulting optimization problems are often not solvable in reasonable time. Therefore, optimizers usually impose assumptions on the function f and the set of feasible solutions X to ensure that a solution can be found or approximated efficiently. These assumptions must be strong enough to guarantee the existence of an efficient algorithm for solving the problem but not too restrictive, so that the framework can still capture a large number of practical problems of interest.

1.2 Large-scale Optimization and First-Order Methods

In this thesis, we focus on unconstrained convex optimization, where the objective function is a closed, proper, convex function, and the feasible set spans the entire space, i.e., $X = \mathbb{R}^d$. We consider the large-scale setting, meaning that the dimension d is very large (larger than the number of iterations of the algorithm used to solve the problem). First-order methods form a class of optimization algorithms particularly well-suited for large-scale problems. These methods generate a sequence of iterates to approximate an optimal solution using only first-order information of the function f . Specifically, assume access to a first-order oracle $\mathcal{O}_f$ that, for any $x \in \mathbb{R}^d$, returns the function value and a subgradient (or the gradient when f is differentiable) at x :

$$\mathcal{O}_f(x) = \{f(x), \partial f(x)\}.$$

A first-order algorithm $\mathcal{M}$ can then be expressed in the following conceptual form:

$$\begin{aligned} x_1 &= \mathcal{M}_1(x_0, \mathcal{O}_f(x_0)), \\ x_2 &= \mathcal{M}_2(x_0, \mathcal{O}_f(x_0), \mathcal{O}_f(x_1)), \\ &\vdots \\ x_N &= \mathcal{M}_N(x_0, \mathcal{O}_f(x_0), \dots, \mathcal{O}_f(x_{N-1})). \end{aligned} \tag{1.1}$$

where $\mathcal{M}_i$ is the operation that outputs the iterate x_i from the initial iterate x_0 and the past first-order information collected about the function f . These methods are particularly appealing for solving large-scale optimization problems because of their a priori low computational cost per iteration compared to more advanced methods [74, 76] that utilize additional information about the function, such as knowledge of higher-order derivatives. While these advanced methods provide more accurate solutions, they come at a high computational cost per iteration. In contrast, first-order methods are significantly more computationally efficient but require a lot of iterations to achieve highly accurate solutions. Thankfully, in most modern applications of large-scale optimization, a highly accurate solution is not required. For instance, in many optimization problems arising in machine learning and signal processing, the objective functions themselves are approximations of the original problem. This has motivated intense research efforts on first-order methods to further improve the efficiency of large-scale opti-

mization. Some key developments in this area include coordinate descent algorithms, which update only one block of variables per iteration [72, 36]; stochastic first-order methods, which use inexpensive approximations of gradient information [16, 5]; and acceleration schemes, which enhance the convergence speed of first-order algorithms [70, 13, 95]. In this thesis, we are interested in developing tools to automatically estimate the efficiency and design optimized first-order algorithms for different classes of large-scale problems.

1.3 Worst-case Analysis of First-Order Methods

In order to evaluate the efficiency of first-order methods and design more efficient algorithms, we first need to define a rigorous framework for the analysis of first-order algorithms. In this thesis, we adopt the worst-case analysis framework. Consider a class of unconstrained optimization problems characterized by a set of assumptions on the objective functions that define a functional class $\mathcal{F}(\mathbb{R}^d)$, a first-order method $\mathcal{M}$, a number of iterations N , the list of iterates $x_1, \dots, x_N$ generated by $\mathcal{M}$ from an initial point x_0 , and a performance criterion $\mathcal{P}(\mathcal{O}_f, x_0, x_1, \dots, x_N, x_*)$, typically:

$$\mathcal{P}(\mathcal{O}_f, x_0, x_1, \dots, x_N, x_*) = f(x_N) - f(x_*),$$

where x_* is a minimizer of f , or

$$\mathcal{P}(\mathcal{O}_f, x_0, x_1, \dots, x_N, x_*) = \|\nabla f(x_N)\|^2,$$

provided that f is assumed to be differentiable. Ensuring that the worst-case performance of a first-order algorithm is bounded usually require to consider initial conditions of the form $\mathcal{I}(f, x_0, x_1, \dots, x_N, x_*) \leq R$. In most cases, it is sufficient to assume that the first iterate x_0 is not "too far" from an optimal point x_* of f which translates into $\mathcal{I}(\mathcal{O}_f, x_0, x_1, \dots, x_N, x_*) = \|x_0 - x_*\|$. The worst-case performance of algorithm $\mathcal{M}$ over the class of problems characterized by $\mathcal{F}(\mathbb{R}^d)$ is then given by:

$$W(\mathcal{F}(\mathbb{R}^d), \mathcal{M}, N, \mathcal{P}, \mathcal{I}) = \sup_{f \in \mathcal{F}(\mathbb{R}^d)} \mathcal{P}(\mathcal{O}_f, x_1, \dots, x_N, x_*)$$

such that $x_1, \dots, x_N$ are generated by $\mathcal{M}$

$$\mathcal{I}(\mathcal{O}_f, x_0, \dots, x_N, x_*) \leq R.$$

Worst-case convergence analysis of algorithm $\mathcal{M}$ reduces to estimating the evolution of its worst-case performance, $W(\mathcal{F}(\mathbb{R}^d), \mathcal{M}, N, \mathcal{P})$, as a function of the number of iterations N . The worst-case performance of an algorithm is expected to decrease with the number of iterations, and the rate at which it decreases with respect to N is referred to as the convergence rate of the algorithm.

Classical methods for worst-case analysis are covered in detail in the books by Nemirovsky and Yudin [69] and Nesterov [71]. A relatively recent approach to worst-case analysis is the Performance Estimation Problem (PEP) framework [33], which will be a central concept for the remainder of this thesis.

1.4 Performance Estimation Problem (PEP)

The recent PEP framework, introduced by Drori and Teboulle [33], provides a systematic approach to performing worst-case convergence analysis for first-order algorithms. Drori and Teboulle observed that, in some cases, the optimization problem defining the worst-case performance of a first-order algorithm admits a tractable relaxation that provides valid upper bounds on the worst-case convergence of the algorithm under analysis. Taylor et al. [97] later introduced the notion of interpolation of finite sets by a class of functions and demonstrated in several cases that Drori and Teboulle's PEP formulation was, in fact, exact. Consider a class of functions $\mathcal{F}(\mathbb{R}^d)$, an initial point x_0 , a first-order algorithm $\mathcal{M}$, a performance criterion $\mathcal{P}$, and a number of iterations N . The worst-case performance of algorithm $\mathcal{M}$ over the functional class $\mathcal{F}(\mathbb{R}^d)$ is given by the infinite-dimensional PEP:

$$\begin{aligned} w(\mathcal{F}, R, \mathcal{M}, N, \mathcal{P}, \mathcal{I}) &= \sup_{f, x_0, x_1, \dots, x_N, x_*} \mathcal{P}(\mathcal{O}_f, x_0, \dots, x_N, x_*), \\ \text{such that } f &\in \mathcal{F}(\mathbb{R}^d), \\ x_* &\text{ is an optimal solution for } f, \\ x_1, \dots, x_N &\text{ are generated from } (f, x_0) \\ &\text{ by method } \mathcal{M} \text{ according to (1.1),} \\ \mathcal{I}(\mathcal{O}_f, x_0, \dots, x_N, x_*) &\leq R, \end{aligned} \tag{i-PEP}$$

where $\mathcal{I}$ imposes an additional constraint ensuring that the worst-case performance is bounded. Infinite-dimensional problems such as (i-PEP)

are typically challenging to solve. To address this, Taylor et al. [97] introduced the concept of $\mathcal{F}(\mathbb{R}^d)$ -interpolability of finite sets, which allows Problem (i-PEP) to be reformulated as a finite-dimensional problem:

Definition 1.1 ($\mathcal{F}(\mathbb{R}^d)$ -interpolability). Let I be an index set, and consider the set of triples $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$, where $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. The set $\mathcal{S}$ is $\mathcal{F}(\mathbb{R}^d)$ -interpolable if and only if there exists a function $f \in \mathcal{F}(\mathbb{R}^d)$ such that $g_i \in \partial f(x_i)$ and $f(x_i) = f_i, \forall i \in I$.

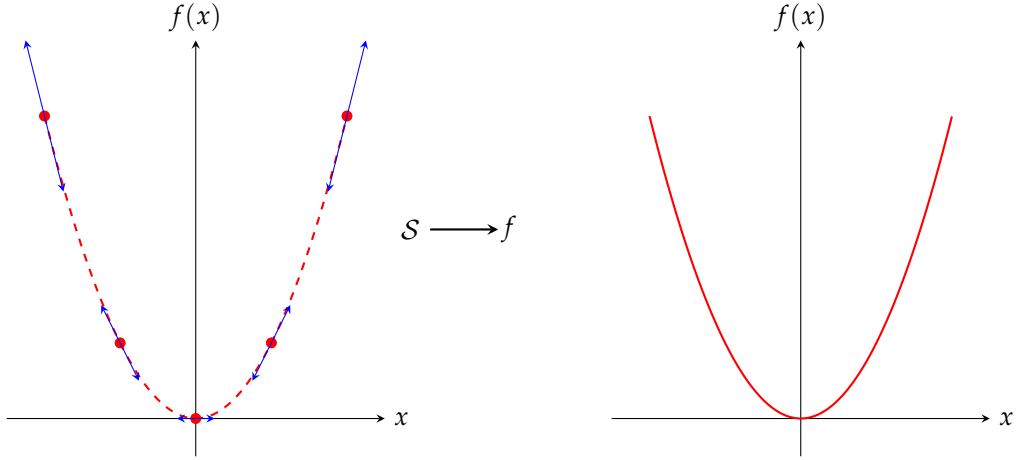


Fig. 1.1 Illustration of the $\mathcal{F}(\mathbb{R}^d)$ -interpolation process of a finite set $\mathcal{S}$.

Since the PEP for a first-order algorithm only involves first-order information of the functional variable f at the iterates x_i of the algorithm and a minimizer x_* , we can replace the functional variable f with a finite set of variables $\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}$ with $I = \{0, 1, \dots, N, *\}$, where $\{x_i\}_{i \in \{0, 1, \dots, N\}}$ are the iterates of the algorithm and x_* is a minimizer of f . By $\mathcal{F}(\mathbb{R}^d)$ -interpolability, there exists a function $f \in \mathcal{F}(\mathbb{R}^d)$ such that, for any $i \in I$, g_i corresponds to the (sub)gradient at x_i and f_i corresponds to the function value at x_i . Thus, optimizing over functions $f \in \mathcal{F}(\mathbb{R}^d)$ is equivalent to optimizing over $\mathcal{F}(\mathbb{R}^d)$ -interpolable sets $\mathcal{S}_N$. Moreover, since we are considering the convex unconstrained minimization setting, the condition that x_* is a minimizer of the functional variable f is equivalent to setting $g_* = 0$. This leads to the following exact finite-dimensional formulation of Problem (i-PEP):

$$\begin{aligned}
w(R, \mathcal{M}, N, \mathcal{P}, \mathcal{I}) = & \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} \mathcal{P}(\mathcal{S}_N), \\
& \text{such that } \mathcal{S}_N \text{ is } \mathcal{F}(\mathbb{R}^d)\text{-interpolable,} \\
& x_1, \dots, x_N \text{ is generated from } x_0 \text{ by method } \mathcal{M} \text{ with (1.1),} \\
& g_* = 0, \\
& \mathcal{I}(\mathcal{S}_N) \leq R.
\end{aligned}$$

(f-PEP)

Now that we have a finite-dimensional formulation for PEPs, the remaining challenge is to express the $\mathcal{F}(\mathbb{R}^d)$ -interpolability constraint on $\mathcal{S}_N$ in a tractable manner. This requires developing necessary and sufficient interpolation conditions, which typically take the form of algebraic inequalities involving the elements of the set $\mathcal{S}_N$. In [97], the authors derived necessary and sufficient conditions for the class of smooth (strongly) convex functions. Deriving necessary interpolation conditions is relatively straightforward: discretizing any valid characterization of the class $\mathcal{F}(\mathbb{R}^d)$ over the set $\mathcal{S}_N$ produces valid necessary conditions. However, proving that necessary interpolation conditions are also sufficient is a highly nontrivial problem. We refer the reader to [86] for more details on this problem. The necessity of interpolation conditions ensures that all sets $\mathcal{S}_N$ corresponding to a function $f \in \mathcal{F}(\mathbb{R}^d)$ are included in the feasible set of the PEP. Furthermore, adding sufficiency guarantees that the feasible set of the PEP is exactly the set of $\mathcal{F}(\mathbb{R}^d)$ -interpolable sets $\mathcal{S}_N$. Consequently, using only necessary interpolation conditions in a PEP reformulation may result in a relaxation, providing only a valid upper bound on the worst-case performance of the algorithm rather than the exact worst-case. Finally, after developing interpolation conditions, the last step is to obtain a tractable (convex) formulation for the PEP. This is usually feasible for the subclass of fixed-step first-order algorithms using a Gram matrix lifting technique, which transforms the problem into a convex Semidefinite Program (SDP) (see [33, 97] for applications to the class of smooth (strongly) convex functions).

Remark 1.2. The PEP methodology can be summarized as follows:

1. Formulate the analysis of the algorithm $\mathcal{M}$ over the class $\mathcal{F}(\mathbb{R}^d)$ as the infinite-dimensional Problem (i-PEP).
2. Convert the infinite-dimensional PEP (i-PEP) into the finite-dimensional PEP (f-PEP) using $\mathcal{F}(\mathbb{R}^d)$ -interpolability (Definition 1.1).

3. Develop explicit necessary (if possible, sufficient) $\mathcal{F}(\mathbb{R}^d)$ -interpolation conditions and impose these conditions on $\mathcal{S}_N$ to require $\mathcal{F}(\mathbb{R}^d)$ -interpolability.
4. Derive a tractable (convex) formulation for the PEP, typically for the subclass of fixed-step first-order algorithms.

The PEP framework has been applied to the worst-case convergence analysis of many first-order algorithms in various settings. In particular, it has been used for the analysis of gradient descent and its accelerated variants on smooth (strongly) convex functions [33, 97], convex functions with quadratic upper bounds [42], and in nonconvex settings [85, 1, 2, 47]. It has also been extended to the analysis of first-order algorithms on smooth strongly convex functions composed with a linear operator [17] and in the relative smoothness setting [29]. Several variants of the gradient descent algorithm have been analyzed using PEP, including gradient descent with exact line search [26], proximal gradient methods [99, 11], and inexact gradient descent with bounded errors in the gradients [102]. Beyond that, PEP frameworks have been developed for several other types of first-order algorithms, including block coordinate descent algorithms [54, 3, 94, 91], stochastic methods [3, 94], adaptive algorithms [25], the difference-of-convex algorithm [4], the Chambolle-Pock algorithm [17], fixed-point iterations [78], algorithms for min-max problems [38], decentralized optimization algorithms [21, 22], subgradient methods [107], and continuous optimization schemes [65]. PEP frameworks have also been developed in the constrained optimization setting for the analysis of the alternating direction method of multipliers [106], in the context of parametric quadratic optimization [81] and for the analysis of Frank-Wolfe and alternating projection methods using a new interpolation theory for convex and strongly convex sets [63]. To enhance accessibility and facilitate worst-case analysis of first-order methods, Python and Matlab toolboxes for PEP have been developed and are available online [40, 98].

PEPs have also been extended to the design of optimized and even optimal first-order methods by introducing an additional layer of minimization over the step sizes of the algorithm within the maximization problem that defines the PEP i.e. using a bilevel formulation. For instance, an optimal first-order algorithm was designed via PEP for smooth convex functions in [55] and for smooth strongly convex functions in [95]. PEPs were also used in [44, 48] to demonstrate that accelerated convergence rates for gradient descent on

smooth convex functions can be achieved solely through careful tuning of the step sizes, without requiring access to past gradient information (as in Nesterov's acceleration). We provide a more detailed review of existing results on the design of optimized algorithms via PEP in Chapter 2.

An alternative approach to PEP for computer-assisted worst-case convergence analysis is the Integral Quadratic Constraints (IQC) framework, proposed in [58], where tools from robust control of nonlinear systems are applied to the analysis of optimization algorithms. In this approach, the optimization algorithm is modeled as a dynamical system, and the nonlinearity arises from the use of gradients to compute the next iterates. The main idea behind IQCs is to model the nonlinearity (the gradient step) "implicitly" by imposing valid quadratic constraints between the inputs and outputs of the dynamical system (the iterative first-order optimization scheme). A convergence rate ρ for the algorithm can then be verified by solving a small SDP, and the optimal convergence rate can be determined using a bisection procedure. This methodology is particularly well-suited for settings where first-order algorithms exhibit linear convergence, such as optimization over smooth strongly convex functions or under the Polyak-Łojasiewicz condition. Similar to PEP, IQCs can also be used to design more efficient algorithms, as demonstrated in [89] for fixed-step first-order methods on smooth strongly convex functions.

Compared to the PEP framework, one advantage of the IQC approach is its tractability, as the corresponding SDPs are typically smaller than those in PEP. However, in contrast to PEP, the IQC approach struggles to handle time-varying step sizes or sublinear convergence rates effectively. Additionally, the IQC framework appears less general, as it requires algorithms to conform to a specific class of linear time-invariant dynamical systems.

1.5 Objectives of the thesis

In this thesis, we first aim to extend the Performance Estimation Problem (PEP) framework for the worst-case analysis of fixed-step first-order algorithms to several classical optimization settings. Specifically, we address block coordinate descent algorithms over coordinate-wise smooth functions (Chapter 3) and gradient descent algorithms over Hölder-smooth convex functions (Chapter 4). We then introduce methods for the numerical design of optimized first-order algorithms (Chapter 5) that are easily applicable

to any optimization setting where PEPs can be formulated as convex conic problems. We demonstrate the flexibility of our algorithm design methods by applying them to several convex optimization settings, including the novel PEP extensions presented in this thesis. In particular, we provide numerical evidence supporting the fact that recent results on the acceleration of first-order methods without using momentum for smooth convex functions [8, 45] can be generalized to the optimization settings considered in this thesis.

1.6 Outline of the Thesis and List of Publications

We begin in Chapter 2 by presenting the PEP framework in greater detail, focusing on the specific case of fixed-step first-order methods applied to smooth (strongly) convex functions. We first review the construction of the convex SDP formulation of PEPs in this setting and summarize key results obtained on the worst-case performance of gradient descent using PEPs. We conclude the first chapter by demonstrating how PEP can simplify the algorithm design problem for fixed-step first-order methods on smooth (strongly) convex functions and reviewing recent advances in the design of optimized fixed-step first-order algorithms.

In Chapter 3, we extend the PEP framework to analyze block coordinate-wise algorithms over coordinate-wise smooth convex functions. Using this extension, we provide new insights into the worst-case behavior of these algorithms, including improved upper bounds on the worst-case performance and a comparison of the efficiency between deterministic and randomized variants.

Next, in Chapter 4, we generalize the PEP framework to analyze fixed-step first-order algorithms over the class of Hölder-smooth convex functions. We introduce a new fixed-step first-order algorithm, provide numerical evidence of its convergence over Hölder-smooth convex functions, and show that our algorithm enjoys a faster convergence rate compared to universal gradient descent when the Hölder exponent is sufficiently close to one.

Finally, in Chapter 5, we address the problem of optimized algorithm design via PEP. We formalize this problem in a general framework and present several numerical methods to solve it. We then numerically design fixed step-size first-order methods in various settings and confirm in several

cases recent findings on optimized first-order methods for smooth convex functions. In particular, we highlight the ability to achieve accelerated convergence rates without using past gradient information (i.e., without momentum).

List of publications.

1. Y. Kamri, F. Glineur and J. M. Hendrickx. *On the minimal and maximal smooth interpolating functions*. In *2021 Benelux Workshop on Systems and Control*, 2021. This paper discusses the pointwise extremal smooth convex interpolating functions of a finite set of triplets. We provide explicit formulas for these functions and show that they admit simpler forms in the one-dimensional case. These results are left outside the scope of this thesis.
2. Y. Kamri, J. M. Hendrickx, and F. Glineur. *On the Worst-Case Analysis of Cyclic Coordinate-Wise Algorithms on Smooth Convex Functions*. In *Proceedings of the 2023 European Control Conference (ECC)*, pages 1–8, 2023. This paper studies the worst-case convergence of block coordinate descent algorithms over the class of smooth convex functions. These results are discussed in Chapter 3.
3. Y. Kamri, F. Glineur, J. M. Hendrickx, and I. Necoara. *On the Worst-Case Analysis of Block Coordinate Descent Algorithms*. *In preparation*. This paper studies the worst-case convergence of block coordinate descent algorithms over the class of coordinate-wise smooth convex functions. These results are presented in Chapter 3.
4. Y. Kamri, J. M. Hendrickx and F. Glineur. *Numerical Design of Optimized First-order Methods*. *In preparation*. This paper studies the worst-case convergence of block coordinate descent algorithms over the class of coordinate-wise smooth convex functions. These results are included in Chapter 5.
5. S. Tanji, Y. Kamri, F. Glineur, M. Madani. *Dual first-order methods for efficient computation of convex hull prices*. *Submitted*, 2025. In this paper, we benchmark a large panel of first-order methods to solve the convex hull pricing problem for electricity markets. We test several methods: a proximal variant of the bundle level method, subgradient methods with three different stepsize strategies, two recent parameter-free methods and an accelerated gradient method combined with smoothing. This work is outside the scope of this thesis.

2

Background: Performance Estimation Problems

2.1 Introduction and Problem Statement

In this chapter, we review fundamental results derived from Performance Estimation Problems (PEPs) concerning the convergence analysis and design of first-order optimization algorithms for the functional classes $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ of L -smooth, μ -strongly convex functions. We begin by defining the functional class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$. For a detailed background on convex analysis, we refer to standard textbooks such as [12, 83, 51]. Let μ and L be two parameters satisfying $0 \leq \mu < L \leq +\infty$. We consider proper, closed, convex functions that satisfy both the L -smoothness and μ -strong convexity properties:

Definition 2.1 (Functional class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$). Consider a proper and closed convex function $f : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$, and constants $\mu \in \mathbb{R}^+$, $L \in \mathbb{R}^+ \cup \{+\infty\}$ with $\mu < L$. We say that function f belongs to $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ if and only if the following two conditions are satisfied:

1. The inequality $\frac{1}{L}\|g_1 - g_2\| \leq \|x_1 - x_2\|$ holds for all pairs $x_1, x_2 \in \mathbb{R}^d$ and corresponding subgradients $g_1 \in \partial f(x_1)$ and $g_2 \in \partial f(x_2)$.
2. The function $f(x) - \frac{\mu}{2}\|x\|^2$ is convex.

When $L = +\infty$, the first condition in Definition 2.1 becomes vacuous. Similarly, when $\mu = 0$, the second condition reduces to the standard convexity of the function f . Hence, the class $\mathcal{F}_{0,+\infty}(\mathbb{R}^d)$ represents the set of proper, closed, convex functions, including nonsmooth functions. For a finite value of L , the first condition in Definition 2.1 implies the uniqueness of the subgradient at every point $x \in \mathbb{R}^d$, ensuring the differentiability of the functions. The class $\mathcal{F}_{0,L}(\mathbb{R}^d)$ then corresponds to the well-known set of L -smooth convex functions. In the case where $\mu > 0$ and $L = +\infty$, we obtain the class of μ -strongly convex functions, denoted by $\mathcal{F}_{\mu,+\infty}(\mathbb{R}^d)$.

For the remainder of this chapter, we focus on the unconstrained optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x),$$

where f belongs to $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$, with parameters satisfying $0 \leq \mu < L < +\infty$. Our objective is to automatically evaluate the efficiency of first-order algorithms, formally defined by (1.1), on this class of problems. Following the PEP methodology summarized in Remark (1.2), the exact worst-case convergence of the first-order algorithm $\mathcal{M}$ over the functional class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ is given by the infinite-dimensional PEP:

$$\begin{aligned} w(\mu, L, R, \mathcal{M}, N, \mathcal{P}, \mathcal{I}) &= \sup_{f, x_0, \dots, x_N, x_*} \mathcal{P}(\mathcal{O}_f, x_0, \dots, x_N, x_*), \\ \text{such that } f &\in \mathcal{F}_{\mu,L}(\mathbb{R}^d), \\ x_* &\text{ is optimal for } f, \\ x_1, \dots, x_N &\text{ is generated from } x_0 \text{ by method } \mathcal{M} \text{ using (1.1),} \\ \mathcal{I}(\mathcal{O}_f, x_0, \dots, x_N, x_*) &\leq R, \end{aligned} \tag{i-PEP-Smooth}$$

For the remainder of this chapter we choose as performance criterion the final objective accuracy

$$\mathcal{P}(\mathcal{O}_f, x_0, \dots, x_N, x_*) = f(x_N) - f(x_*),$$

and, as initial condition to ensure that the worst-case of the algorithm is bounded, the initial distance to a solution

$$\mathcal{I}(\mathcal{O}_f, x_0, \dots, x_N, x_*) = \|x_0 - x_*\|.$$

This problem can be reduced to a finite-dimensional formulation using the

notion of interpolability, which is defined for an arbitrary class of functions in Definition (1.1) and specialized for the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ in the following definition:

Definition 2.2 ($\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolability). Let I be an index set, and consider the set of triples $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$, where $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. The set $\mathcal{S}$ is $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolable if and only if there exists a function $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ such that $g_i \in \partial f(x_i)$ and $f(x_i) = f_i$, $\forall i \in I$.

As explained in Section 1.4 for an arbitrary class of functions, thanks to $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolability, the functional variable $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ in Problem (i-PEP-Smooth) can be replaced by a finite $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolable set of decision variables $\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I} \in (\mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R})^{N+2}$ with $I = \{0, 1, \dots, N, *\}$ such that Problem (i-PEP-Smooth) takes the following finite-dimensional form:

$$\begin{aligned}
w(\mu, L, R, \mathcal{M}, N) = & \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} f_N - f_*, \\
& \text{such that } \mathcal{S}_N \text{ is } \mathcal{F}_{\mu,L}(\mathbb{R}^d)\text{-interpolable,} \\
& x_1, \dots, x_N \text{ is generated from } x_0 \text{ by } \mathcal{M} \text{ with (1.1),} \\
& g_* = 0, \\
& \|x_0 - x_*\| \leq R.
\end{aligned}$$

(f-PEP-Smooth)

In Section 2.2 of this chapter, we review necessary and sufficient interpolation conditions for $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolability. In Section 2.3, we provide a convex reformulation of Problem (f-PEP-Smooth) for fixed-step first-order algorithms. In Section 2.4, as an illustrative application of PEP, we review fundamental results on the convergence of gradient descent over the class of L -smooth convex functions. Building upon this, we then examine how this convex PEP formulation can be leveraged to explore the design of optimized first-order algorithms. Conceptually, an optimal first-order algorithm over the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ —one that minimizes the worst-case performance over the class—solves the following infinite-dimensional min-max problem:

$$w^{\text{opt}}(\mu, L, R, N) = \inf_{\mathcal{M}} \sup_{f, x_0, \dots, x_N, x_*} f(x_N) - f(x_*),$$

such that:

$$f \in \mathcal{F}_{\mu, L}(\mathbb{R}^d),$$

x_* is optimal for f ,

$x_1, \dots, x_N$ is generated from x_0 by method $\mathcal{M}$ with $\mathcal{O}_f$,

$$\|x_0 - x_*\| \leq R.$$

(i-opt-PEP-Smooth)

Note that in Problem (i-opt-PEP-Smooth), the inner maximization problem corresponds to the PEP of the algorithm and, as explained previously, can be transformed into a finite-dimensional problem:

$$w^{\text{opt}}(\mu, L, R, N) = \inf_{\mathcal{M}} \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} f_N - f_*,$$

such that $\mathcal{S}_N$ is $\mathcal{F}_{\mu, L}(\mathbb{R}^d)$ -interpolable,

$x_1, \dots, x_N$ is generated from x_0 by method $\mathcal{M}$ with (1.1),

$$g_* = 0,$$

$$\|x_0 - x_*\| \leq R.$$

(f-opt-PEP-Smooth)

In Section 2.5, we demonstrate how Problem (f-opt-PEP-Smooth) can be simplified into a tractable, albeit nonconvex, formulation for fixed-step first-order methods. We review various approaches to solving this problem and highlight several recent algorithms derived from these solutions. Specifically, we discuss the development of optimal fixed-step first-order algorithms for the smooth convex and smooth strongly convex settings, as well as optimized gradient descent algorithms that achieve accelerated convergence rates without relying on past gradient information (i.e., momentum-based methods such as Nesterov's acceleration).

2.2 Smooth (Strongly)-Convex Functions

As mentioned in the introduction of this chapter, our goal is to review interpolation conditions that characterize the $\mathcal{F}_{\mu, L}(\mathbb{R}^d)$ -interpolability of a finite set, allowing us to reformulate the infinite-dimensional PEP (i-PEP-Smooth) for this class as a tractable finite-dimensional problem (f-PEP-Smooth). Nec-

essary interpolation conditions for a given class of functions can be derived by discretizing any condition that holds for the class over the entire domain. To this end, we present well-known characterizations of the function classes $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$:

Theorem 2.3. [71, Theorem 2.1.5] Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ a differentiable function. We have $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ if and only if for all $x_1, x_2 \in \mathbb{R}^d$:

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle, \quad (2.1)$$

$$f(x_2) \leq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{L}{2} \|x_1 - x_2\|^2. \quad (2.2)$$

Theorem 2.4. [71, Theorem 2.1.5] Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ a differentiable function. We have $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ if and only if for all $x_1, x_2 \in \mathbb{R}^d$:

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{1}{2L} \|\nabla f(x_1) - \nabla f(x_2)\|^2. \quad (2.3)$$

Theorem 2.5. Let $f : \mathbb{R}^d \mapsto \mathbb{R}$. Then $f \in \mathcal{F}_{\mu,\infty}(\mathbb{R}^d)$ if and only if for all $x_1, x_2 \in \mathbb{R}^d$, and for all $g_1 \in \mathbb{R}^d$ such that $g_1 \in \partial f(x)$

$$f(x_2) \geq f(x_1) + \langle g_1, x_2 - x_1 \rangle + \frac{\mu}{2} \|x - y\|^2, \quad (2.4)$$

Discretizing any of the conditions given in the previous three theorems provides necessary interpolation conditions for the corresponding function class. However, the resulting conditions are not necessarily sufficient. We will clarify this point after stating the necessary and sufficient interpolation conditions for the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$:

Theorem 2.6 ($\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolability). [97, Theorem 4]. Given a finite index set I . A set $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$ is $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolable if and only if the following set of conditions holds for every pair of indices $i \in I$ and $j \in I$:

$$f_i - f_j - \langle g_j, x_i - x_j \rangle \geq \frac{1}{2(1 - \mu/L)} \left(\frac{1}{L} \|g_i - g_j\|^2 + \mu \|x_i - x_j\|^2 - \frac{2\mu}{L} \langle g_j - g_i, x_j - x_i \rangle \right). \quad (2.5)$$

Note that when $\mu = 0$, the interpolation conditions in (2.5) reduce to the discretization of the conditions in Theorem 2.4.

Lemma 2.7. (L -smooth convex interpolability). Given a finite index set I . A set $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$ is $\mathcal{F}_{0,L}(\mathbb{R}^d)$ -interpolable if and only if the following set of

conditions holds for every pair of indices $i \in I$ and $j \in I$:

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \frac{1}{2L} \|g_i - g_j\|^2. \quad (2.6)$$

Similarly, when $L = +\infty$ and using the convention $1 / +\infty = 0$, these conditions reduce to the discretization of the conditions in Theorem 2.5.

Lemma 2.8. (*μ -strongly convex interpolability*). *Given a finite index set I . A set $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$ is $\mathcal{F}_{\mu, +\infty}(\mathbb{R}^d)$ -interpolable if and only if the following set of conditions holds for every pair of indices $i \in I$ and $j \in I$:*

$$f_i \geq f_j + \langle g_j, x_i - x_j \rangle + \frac{\mu}{2} \|x_i - x_j\|^2 \quad (2.7)$$

Remark 2.9. Discretizing the conditions in Theorem 2.3 provides necessary but not sufficient interpolation conditions for the class $\mathcal{F}_{0,L}(\mathbb{R}^d)$. Consider the following counterexample from [97]:

$$(x_1, g_1, f_1) = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, 0 \right) \quad \text{and} \quad (x_2, g_2, f_2) = \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, 1 \right),$$

This set satisfies the conditions of Theorem 2.3 with $L = 1$ but cannot be interpolated by a smooth convex function. The convexity condition forces the interpolating function to lie above its linear under-approximations, leading to non-differentiability at both x_1 and x_2 .

This example demonstrates that discretizing necessary and sufficient characterizations of a function class does not necessarily yield sufficient interpolation conditions. Consequently, this can introduce a relaxation in the finite reformulation of the PEP, resulting in only upper bounds on the algorithm's worst-case performance.

2.3 Convex Formulation of the Performance Estimation Problem

Building on the interpolation results from the previous section, we now provide a finite-dimensional convex formulation of the Performance Estimation Problem for the functional class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$, following Section 3 of [97]. As outlined in the introduction of this chapter, the Performance Estimation Problem admits a finite-dimensional representation. Specifically, the

functional decision variable $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ is replaced by a set of decision variables $\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}$, where $I = \{0, \dots, N, *\}$. For all $i \in I$, the variables $g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ are constrained by the $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolation conditions to be the gradient and function value at x_i for some function $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$. The variables x_i for $i \in \{0, \dots, N\}$ represent the first N iterates of algorithm $\mathcal{M}$, and x_* is enforced to be an optimal point of f by the condition $g_* = 0$. Since the functional class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ is invariant under additive shifts in function values and translations in the domain, we can assume without loss of generality that $x_* = 0$ and $f_* = 0$. The Performance Estimation Problem can then be formulated in the following finite-dimensional form:

$$\begin{aligned}
w^{(d)}(\mu, L, R, \mathcal{M}, N, \mathcal{P}) &= \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} f_N - f_*, \\
\forall i, j \in I, \quad &f_i - f_j - \langle g_j, x_i - x_j \rangle \geq \\
&\frac{1}{2(1 - \mu/L)} \left(\frac{1}{L} \|g_i - g_j\|^2 + \mu \|x_i - x_j\|^2 - \frac{2\mu}{L} (g_j - g_i)^\top (x_j - x_i) \right) \\
&x_1, \dots, x_N \text{ is generated from } x_0 \text{ by method } \mathcal{M} \text{ with (1.1),} \\
&\{x_*, g_*, f_*\} = \{0, 0, 0\}, \\
&\|x_0 - x_*\| \leq R.
\end{aligned}$$

(d-PEP-Smooth)

Problem (d-PEP-Smooth) is an exact reformulation of (f-PEP-Smooth), where the $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolability constraint on the sets $\mathcal{S}_N$ is replaced by the necessary and sufficient interpolation conditions from Theorem 2.6. Note that the Performance Estimation Problem in the form of Problem (d-PEP-Smooth) is nonconvex, as it includes nonconvex quadratic constraints (e.g., terms like $g_j^\top x_i$ in the interpolation conditions). We now introduce the specific class of fixed-step first-order algorithms:

Definition 2.10. (Fixed-step first-order method). An algorithm $\mathcal{M}$ is called a *fixed-step first-order method* if its iterates are computed as

$$x_i = x_0 - \sum_{k=0}^{i-1} \alpha_{i,k} g_k, \quad (2.8)$$

with fixed scalar coefficients $\alpha_{i,k}$. We denote this method by $\mathcal{M}_\alpha$.

For this class of methods, Problem (d-PEP-Smooth) can be reformulated as a convex semidefinite program (SDP) using a Gram matrix lifting. Consider the matrix formed by grouping the gradients at each iterate and the initial

point x_0 ($N + 2$ columns):

$$P = [g_0, g_1, \dots, g_N, x_0], \quad (2.9)$$

and the Gram matrix $G = P^\top P \in \mathbb{S}^{N+2}$. Any linear expression involving products of iterates and their associated gradients can be fully represented using the matrix G . Define the vectors $h_i \in \mathbb{R}^{N+2}$ for $i \in \{0, \dots, N\}$ and h_* as:

$$h_i^\top = [-\alpha_{i,0} \ -\alpha_{i,1} \ \dots \ -\alpha_{i,i-1} \ 0 \ \dots \ 0 \ 1], \quad h_*^\top = [0 \ \dots \ 0], \quad (2.10)$$

so that for all $i \in \{0, \dots, N\}$, we have $x_i = Ph_i$ and $x_* = Ph_*$. Similarly, let $u_i = e_{i+1} \in \mathbb{R}^{N+2}$, where $\{e_i\}_{i \in \{1, \dots, N+2\}}$ is the canonical basis of $\mathbb{R}^{N+2}$, and define $u_* = 0$ to represent the gradients, such that $g_i = Pu_i$ and $g_* = Pu_*$. Given these definitions, the following holds:

$$\begin{aligned} \forall i, j \in I, \quad \langle x_i, x_j \rangle &= h_i^\top G h_j, \\ \langle g_i, g_j \rangle &= u_i^\top G u_j, \\ \langle g_i, x_j \rangle &= u_i^\top G h_j. \end{aligned}$$

Hence, the constraints of Problem (d-PEP-Smooth) can be expressed using only the Gram matrix G and the vector of function values $F^\top = [f_0, \dots, f_N]$. For the interpolation constraints, we can now write:

$$\begin{aligned} F^\top(u_i - u_j) &\geq \frac{L}{L - \mu} \left(u_j^\top G h_i - u_j^\top G h_j \right) + \frac{1}{2(L - \mu)} (u_i - u_j)^\top G (u_i - u_j) \\ &\quad + \frac{\mu}{L - \mu} \left(u_i^\top G h_j - u_i^\top G h_i \right) \\ &\quad + \frac{L\mu}{2(L - \mu)} (h_i - h_j)^\top G (h_i - h_j), \quad \forall i, j \in I. \end{aligned}$$

We can express this more compactly using the trace operator by defining the matrices:

$$\begin{aligned} 2A_{ij}(\alpha) &= \frac{L}{L - \mu} \left(u_j(h_i - h_j)^\top + (h_i - h_j)u_j^\top \right) + \frac{1}{L - \mu} (u_i - u_j)(u_i - u_j)^\top \\ &\quad + \frac{\mu}{L - \mu} \left(u_i(h_j - h_i)^\top + (h_j - h_i)u_i^\top \right) \\ &\quad + \frac{L\mu}{L - \mu} (h_i - h_j)(h_i - h_j)^\top, \quad \forall i, j \in I. \end{aligned}$$

The interpolation conditions can then be written as:

$$F^\top(u_j - u_i) + \text{Tr}(A_{ij}(\alpha)G) \leq 0.$$

The initial condition $\|x_0 - x_*\|^2 \leq R^2$ can be expressed as:

$$\text{Tr}(A_R G) \leq R^2,$$

with $A_R = (u_{N+1} - u_*)(u_{N+1} - u_*)^\top$. For the performance criterion $\mathcal{P}$, any linear criterion of the form

$$\mathcal{P}(F, G) = b^\top F + \text{Tr}(CG),$$

can be incorporated into a convex SDP. This includes common criteria such as $f(x_N) - f_*$, $\|\nabla f(x_N)\|^2$, and $\|x_N - x_*\|^2$. For our criterion of choice, we have $f_N - f_* = b^\top F$ with $b = (u_N - u_*)$. We can now formulate the Performance Estimation Problem for the fixed-step first-order method $\mathcal{M}_\alpha$ over the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ as:

$$\begin{aligned} & \sup_{F \in \mathbb{R}^{N+1}, G \in \mathbb{R}^{(N+2) \times (N+2)}} && b^\top F \\ & \text{subject to:} && F^\top(u_j - u_i) + \text{Tr}(A_{ij}(\alpha)G) \leq 0, \quad \forall i, j \in I, \\ & && \text{Tr}(A_R G) \leq R^2, \\ & && G \succeq 0, \\ & && \text{rank}(G) \leq d. \end{aligned} \quad (\text{d-SDP-PEP-Smooth})$$

Remark 2.11. Note that Problem (d-SDP-PEP-Smooth) is an exact reformulation of the Performance Estimation Problem (i-PEP-Smooth). Consider any d -dimensional function $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ which is a feasible point of Problem (i-PEP-Smooth). Applying the algorithm $\mathcal{M}_\alpha$ to f generates the iterates x_i , their associated gradients g_i , and function values f_i . The set $\{x_i, g_i, f_i\}_{i \in I}$ is clearly $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolable. By constructing the matrix G and vector F , we obtain a matrix G of rank at most d , since the vectors x_i and g_i belong to $\mathbb{R}^d$. Thus, (F, G) forms a feasible point of Problem (d-SDP-PEP-Smooth). Conversely, given any feasible point (F, G) of Problem (d-SDP-PEP-Smooth), where G has rank at most d , one can reconstruct a set $\{x_i, g_i, f_i\}_{i \in I}$ with $x_i, g_i \in \mathbb{R}^d$ using a Cholesky decomposition of G . By the interpolation conditions, the set $\{x_i, g_i, f_i\}_{i \in I}$ is $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -

interpolable. Therefore, there exists a d -dimensional function $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ such that the vectors x_i correspond to the iterates of the algorithm $\mathcal{M}_\alpha$, with the associated gradients g_i and function values f_i .

Note that the matrix G in Problem (d-SDP-PEP-Smooth) has size $(N+2) \times (N+2)$, which implies that its rank is at most $N+2$. Consequently, when $d \geq N+2$, the nonconvex rank constraint in Problem (d-SDP-PEP-Smooth) can be omitted, resulting in a convex SDP formulation for the PEP. The case $d \geq N+2$ corresponds to the large-scale regime, where the function's dimension exceeds the number of algorithm steps. This setting is particularly relevant in most modern applications of mathematical optimization.

We now present an exact convex SDP formulation for the PEPs of fixed-step algorithms over the function class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$:

Theorem 2.12. ([97, Theorem 5]) *Consider the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ of L -smooth μ -strongly convex functions with $0 \leq \mu < L \leq \infty$, the fixed-step first-order method $\mathcal{M}_\alpha$. If $d \geq N+2$, the worst-case performance after N iterations of method $\mathcal{M}_\alpha$ over the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ is equal to the optimal value of the following semidefinite program:*

$$\begin{aligned} w^{\text{SDP}}(\mu, L, R, \alpha_{i,k}, N, b) = & \sup_{G \in \mathbb{S}^{N+2}, f \in \mathbb{R}^{N+1}} b^\top F \\ & \text{such that:} \\ & F^\top(u_j - u_i) + \text{Tr}(A_{i,j}(\alpha)G) \leq 0, \quad i, j \in I, \\ & \text{Tr}(A_R G) - R^2 \leq 0, \\ & G \succeq 0, \end{aligned} \tag{SDP-PEP-Smooth}$$

with matrices $A_{i,j}(\alpha)$, A_R and vector b as defined above.

Remark 2.13. Since the convex PEP formulation, Problem (SDP-PEP-Smooth), is independent on the dimension d , it also provides the exact-worst case performance of the fixed-step first-order algorithm, $\mathcal{M}_\alpha$, over the set $\mathcal{F}_{\mu,L} = \bigcup_{d \in \mathbb{N}} \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ of all L -smooth μ -strongly convex functions.

Remark 2.14. A limitation of the PEP framework is that the number of iterations for the algorithm is a parameter of Problem (SDP-PEP-Smooth) that needs to be specified beforehand. Indeed, the dimension of the semidefinite variable G in Problem (SDP-PEP-Smooth) is equal to $N+2$. Thus, computing the worst-case convergence rate of a first-order method with respect to the number of iterations N requires solving multiple SDPs of

increasing size and at higher precisions since the worst-case is expected to decrease with the number of iterations N . Solving semidefinite programs of increasing size using standard interior-point solvers usually becomes computationally expensive and memory intensive, making them suitable only for small to medium-scale problems (up to around $N = 100$ for PEPs). Moreover, it is not clear whether PEP formulations exhibit any particular exploitable structure to make their resolution more efficient (the matrix G in Problem (SDP-PEP-Smooth) is a priori dense).

When $L < +\infty$, the optimal value of Problem (SDP-PEP-Smooth) is finite and is attained at a feasible point (see [97, Proposition 1]). In the case $L = +\infty$ (nonsmooth convex functions), additional constraints such as $\|g_i\| \leq M$, which enforce a Lipschitz condition on the function, are required to ensure a finite solution to Problem (SDP-PEP-Smooth) (see [97], Remark 4). For certain classical performance criteria, Problem (SDP-PEP-Smooth) exhibits homogeneity with respect to the smoothness constant L and the initial condition bound R , provided that the step sizes of the algorithm are normalized by a factor of $1/L$. Thus, for such algorithms, solving Problem (SDP-PEP-Smooth) once with $L = 1$ and $R = 1$ is sufficient to deduce the worst-case performance for any values of L and R . For our functional accuracy criterion, we have:

Theorem 2.15. (see [96, Section 4.2.5].) *The following inequality hold:*

$$w^{SDP}(\mu, L, R, \frac{\alpha}{L}, N, b) = LR^2 w^{SDP}(q, 1, 1, \alpha, N, b)$$

where $q = \frac{\mu}{L}$ is the inverse condition number.

In general, finding a closed-form solution to Problem (SDP-PEP-Smooth) is challenging. Even if such a solution is identified, Problem (SDP-PEP-Smooth) does not provide an analytical proof for the derived worst-case performance. These limitations can be overcome by considering the Lagrangian dual of

Problem (SDP-PEP-Smooth) (see [97, Section 3.4]):

$$\begin{aligned}
& \inf_{\lambda_{ij}, \tau} \tau R^2 \\
& \text{such that:} \\
& \tau A_R + \sum_{i,j \in I} \lambda_{ij} A_{i,j}(\alpha) \succeq 0, \\
& b - \sum_{i,j \in I} \lambda_{ij} (u_j - u_i) = 0, \\
& \lambda_{ij} \geq 0, \quad i, j \in I, \\
& \tau \geq 0.
\end{aligned} \tag{dual-SDP-PEP-Smooth}$$

$\lambda_{i,j}$ are the dual variables associated to the interpolation conditions and τ is the dual variable associated to the initial condition. Under the mild condition $\alpha_{i,i-1} \neq 0$ (i.e., the algorithm uses the gradient at the latest iterate for the next update), the duality gap is zero.

Theorem 2.16. ([97, Theorem 6.1]) *The optimal value of (dual-SDP-PEP-Smooth) with $0 \leq \mu < L < \infty$ is attained and equal to $w^{SDP}(R, \mu, L, \alpha, N, b, C)$ under the assumptions that $\alpha_{i,i-1} \neq 0$, for all $i \in \{1, \dots, N-1\}$.*

Remark 2.17. Since the duality gap is zero under the assumptions of Theorem 2.16, solving Problem (dual-SDP-PEP-Smooth) not only provides an alternative approach to computing the exact worst-case performance of the algorithm but also offers a formal proof to verify the worst-case bound. Informally, the worst-case bound is obtained by multiplying each constraint of Problem (SDP-PEP-Smooth) by the corresponding dual variables $\lambda_{i,j}$ and then summing them. For a more detailed and rigorous presentation of PEP-generated proofs, we refer the reader to [39].

2.4 Use Case Example of PEP: Worst-case Analysis of Gradient Descent for L -Smooth Convex Functions

After introducing the Performance Estimation framework for the function classes $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ in the previous section, we now illustrate its application by reviewing worst-case analysis results for gradient descent obtained using this framework. Gradient descent is one of the most fundamental and widely used optimization methods, dating back to the work of Cauchy [20]:

Algorithm 1 Gradient Method with constant step-sizes (GM)

```
1: Input:  $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ ,  $x_0 \in \mathbb{R}^d$ ,  
2: for  $i = 0$  to  $N - 1$  do  
3:    $x_{i+1} = x_i - \frac{\alpha}{L} \nabla f(x_i)$   
4: end for
```

We consider a normalized step size α/L to exploit the homogeneity properties of the PEP framework (see Theorem 2.15). Since gradient descent is a fixed-step first-order method, it can be expressed in the form of (1.1), making it suitable for PEP-based analysis. Using a relaxation of Problem (SDP-PEP-Smooth), which was later proven to be exact, Drori and Teboulle [33] conjectured the following result on the worst-case performance of gradient descent:

Theorem 2.18. [33, Conjecture 3.1] *After N steps of (GM) with constant normalized step size $0 \leq \alpha \leq 2$ on a smooth convex function $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, the following holds*

$$f(x_N) - f(x_*) \leq \frac{LR^2}{2} \max \left(\frac{1}{2N\alpha + 1}, (1 - \alpha)^{2N} \right). \quad (2.11)$$

The bound in Theorem 2.18 cannot be improved, as it is attained by gradient descent on two one-dimensional L -smooth convex functions:

$$\begin{aligned} f_1(x) &= \begin{cases} \frac{LR}{2N\alpha+1}|x| - \frac{LR^2}{2(2N\alpha+1)^2}, & \text{if } |x| \geq \frac{R}{2N\alpha+1}, \\ \frac{L}{2}x^2, & \text{else,} \end{cases} \\ f_2(x) &= \frac{L}{2}x^2. \end{aligned} \quad (2.12)$$

A proof of the result for $0 \leq \alpha \leq 1$ was provided in [33] by deriving an analytical solution to the dual formulation of the PEP for the gradient descent algorithm, using technical and non-standard arguments. Later, Teboulle and Vaisbourd [100] presented an alternative proof for the same case, $0 \leq \alpha \leq 1$, which was more natural in the context of convex optimization. Their approach relied on a new descent lemma (see [100], Lemma 1). The case $1 < \alpha < 2$ remains an open problem (see however the recent preprint [57]). This conjecture also suggests a method for determining the optimal fixed step size for gradient descent after N iterations. The optimal step size α_{opt} minimizes the right-hand side of the bound (2.11) and is computed by solving a nonlinear equation, for which a closed-form solution appears difficult to obtain:

$$\frac{1}{2N\alpha_{\text{opt}} + 1} = (1 - \alpha_{\text{opt}})^{2N}. \quad (2.13)$$

This balances the performance of gradient descent across both functions f_1 and f_2 . The optimal step size α_{opt} ensures fast convergence on f_1 while avoiding overshooting on the quadratic function f_2 . Recent progress on the conjecture regarding the optimal fixed step size was made by Grimmer et al. [46]. Similarly, using PEP, Rotaru et al. [85] conducted a comprehensive analysis of gradient descent with the residual gradient norm as the performance criterion. Their study provided exact analytical worst-case results for various function classes including smooth convex, strongly convex and weakly convex functions, and examined optimal step sizes in these cases.

2.5 Design of Optimal First-Order Algorithms for the Functional class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$

As outlined in the introduction of this chapter, the Performance Estimation framework provides a systematic approach to formulating algorithm design problems (see Problems (i-opt-PEP-Smooth) and (f-opt-PEP-Smooth)). In this section, we present a simplified yet exact formulation of the algorithm design problem for fixed-step first-order algorithms over the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$. We also briefly discuss existing strategies for solving this problem and highlight optimized algorithms obtained using this approach. We begin by simplifying the algorithm design problem, transforming it from a min-max problem into a minimization problem. In Problem (f-opt-PEP-Smooth), the inner maximization represents the finite-dimensional formulation of the PEP. For fixed-step first-order algorithms over the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$, we replace this inner maximization with the Lagrangian dual of the PEP, given by Problem (dual-SDP-PEP-Smooth).

$$\begin{aligned} w^{\text{opt}}(\mu, L, R, N) &= \inf_{\alpha_{i,k}, \lambda_{ij}, \tau} \tau R^2 \\ &\text{such that:} \\ A(\lambda, \alpha) &\succeq 0, \\ b - \sum_{i,j \in I} \lambda_{ij}(u_j - u_i) &= 0, \\ \lambda_{ij} &\geq 0, \quad i, j \in I, \\ \tau &\geq 0, \end{aligned} \quad (\text{opt-PEP})$$

with $A(\lambda, \alpha) = \tau A_R - C + \sum_{i,j \in I} \lambda_{ij} A_{ij}(\alpha)$.

Remark 2.19. Under mild conditions, the duality gap for PEP is zero (see Theorem 2.16). Thus, Problem (opt-PEP) provides an exact reformulation of Problem (f-opt-PEP-Smooth) for fixed-step first-order algorithms.

Remark 2.20. The matrices A_{ij} are linear in the variables $\alpha_{i,k}$ when $\mu = 0$ (for L -smooth convex functions), as the vectors h_i depend linearly on $\alpha_{i,k}$ (see Section 2.3). For $\mu > 0$ (in the L -smooth, μ -strongly convex case), the matrices become bilinear in the variables $\alpha_{i,k}$. Consequently, the matrix $A(\lambda, \alpha)$ is at least bilinear in the variables λ_{ij} and $\alpha_{i,k}$, making the problem nonconvex. In general, problems with bilinear matrix inequalities are NP-hard [101].

A common approach to solving optimization problems with nonlinear matrix inequalities is to derive specific SDP relaxations for the problem. In [55, Section 6.2], Kim and Fessler developed such a relaxation for the function class of L -smooth convex functions, $\mathcal{F}_{0,L}(\mathbb{R}^d)$. They successfully found an analytical solution to their relaxed problem, leading to the Optimized Gradient Method (OGM):

Algorithm 2 Optimized Gradient Method (OGM)

```

1: Input:  $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ ,  $x_0 \in \mathbb{R}^d$ ,  $y_0 = x_0$ ,  $\theta_0 = 1$ 
2: for  $i = 0, \dots, N-1$  do
3:    $y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$ 
4:    $\theta_{i+1} = \begin{cases} \frac{1+\sqrt{1+4\theta_i^2}}{2}, & \text{if } i \leq N-2 \\ \frac{1+\sqrt{1+8\theta_i^2}}{2}, & \text{if } i = N-1 \end{cases}$ 
5:    $x_{i+1} = y_{i+1} + \frac{\theta_i-1}{\theta_{i+1}}(y_{i+1} - y_i) + \frac{\theta_i}{\theta_{i+1}}(y_{i+1} - x_i)$ 
6: end for
```

Note that (OGM) admits a relatively simple structured form, even though the definition of fixed-step first-order algorithms allows for a vast search space, including algorithms with very complicated expressions. Also, (OGM) closely resembles Nesterov's fast gradient method (FGM), the first practical method to achieve the optimal convergence rate of $O(1/N^2)$ for smooth convex functions [70].

Algorithm 3 Fast Gradient Method (FGM)

```
1: Input:  $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ ,  $x_0 \in \mathbb{R}^d$ ,  $y_0 = x_0$ ,  $\theta_0 = 1$ 
2: for  $i = 0, \dots, N - 1$  do
3:    $y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$ 
4:    $\theta_{i+1} = \frac{1 + \sqrt{1 + 4\theta_i^2}}{2}$ 
5:    $x_{i+1} = y_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}} (y_{i+1} - y_i)$ 
6: end for
```

(OGM) accelerates the convergence of (FGM) by a multiplicative constant factor of 2 by adding the extra momentum term $\frac{\theta_i}{\theta_{i+1}} (y_{i+1} - x_i)$:

Theorem 2.21. *Let $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ and let $y_0, \dots, y_N \in \mathbb{R}^d$ be generated by (OGM). Then:*

$$f(y_N) - f(x_*) \leq \frac{LR^2}{4\theta_N^2 - 1} \leq \frac{LR^2}{(N+1)^2}.$$

In [30], Drori established a matching lower bound, proving that (OGM) is optimal—no first-order method can achieve a lower worst-case bound for minimizing smooth convex functions. In [95, Section 3], Taylor and Drori addressed the case of L -smooth, μ -strongly convex functions by applying a reparametrization that transforms the algorithm design problem for the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ into one for the class $\mathcal{F}_{0,L-\mu}(\mathbb{R}^d)$ (see [95, Lemma 3]). This reformulation reduces the problem to only bilinear matrix inequalities (see Remark 2.20). They then developed a similar SDP relaxation and introduced the ITEM method:

Algorithm 4 Information-Theoretic Exact Method (ITEM)

```
1: Input:  $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$  with  $0 \leq \mu < L < \infty$ , initial guess  $x_0 \in \mathbb{R}^d$ 
2: Initialization:  $y_{-1} = z_0 = x_0$ ,  $A_0 = 0$ ,  $q = \mu/L$ 
3: for  $i = 0, 1, \dots, N$  do
4:   Set  $A_{i+1} = \frac{(1+q)A_i + 2(1 + \sqrt{(1+A_i)(1+qA_i)})}{(1-q)^2}$ 
5:    $\beta_i = \frac{A_i}{(1-q)A_{i+1}}$ ,  $\delta_i = \frac{1}{2} \frac{(1-q)^2 A_{i+1} - (1+q)A_i}{1+q+qA_i}$ 
6:    $y_i = (1 - \beta_i)z_i + \beta_i x_i$ 
7:    $x_{i+1} = y_i - \frac{1}{L} \nabla f(y_i)$ 
8:    $z_{i+1} = (1 - q\delta_i)z_i + q\delta_i y_i - \frac{\delta_i}{L} \nabla f(y_i)$ 
9: end for
```

The performance of (ITEM) is given by the following theorem:

Theorem 2.22. [95, Theorem 3] Let $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ and denote $q = \mu/L$. For any $x_0 = z_0 \in \mathbb{R}^d$ and $N \in \mathbb{N}$ with $N \geq 1$, the iterates of (ITEM) satisfy

$$\begin{aligned} \|z_N - x_*\|^2 &\leq \frac{1}{1 + qA_N} \|z_0 - x_*\|^2 \leq \frac{(1 - \sqrt{q})^{2N}}{(1 - \sqrt{q})^{2N} + q} \|z_0 - x_*\|^2, \\ \psi_N &\leq \frac{L}{(1 - q)A_{N+1}} \|z_0 - x_*\|^2 \\ &\leq \min \left\{ (1 - \sqrt{q})^{2(N+1)}, \frac{1}{(N+1)^2} \right\} \frac{L}{(1 - q)} \|z_0 - x_*\|^2, \end{aligned}$$

with $\psi_N = f(y_N) - f_* - \frac{1}{2L} \|\nabla f(y_N)\|^2 - \frac{\mu}{2(1-\mu/L)} \|y_N - \frac{1}{L} \nabla f(y_N) - x_*\|^2 \geq 0$.

The quantity ψ is related to a Lyapunov function that is key to their analysis of the method. Note that the three sequences generated by (ITEM) are valid approximations of the solution x_* in the sense that z_N converges in terms of distance to the optimal solution, y_N is guaranteed to have a small corresponding ψ_N , and x_N corresponds to having a guarantee on the convergence of $f(x_N) - f_*$ in the smooth convex case $\mu = 0$. In [95, Section 2.2], Taylor and Drori analyze two limiting cases of (ITEM). When $\mu = 0$, (ITEM) reduces to the optimal method for smooth convex functions, (OGM). In the case where $\mu > 0$ and $i \rightarrow +\infty$, the parameters of (ITEM) converge to those of the Triple Momentum Method, developed in [89] using the IQC methodology (see [58] for details). They also prove the optimality of (ITEM) for the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ by establishing that the upper bound in Theorem 2.22 matches the lower bound from [31].

In [32], Drori and Taylor propose an alternative approach for designing optimized first-order methods using performance estimation. They begin by formulating a tractable PEP framework tailored to the Greedy First-Order algorithm, which is defined as follows (see [32, Section 3] for details):

Algorithm 5 Greedy First-Order Method (GFM)

Require: $f : \mathbb{R}^d \mapsto \mathbb{R}$ a convex function, initial iterate $x_0 \in \mathbb{R}^d$.

```
1: for  $i = 1, 2, \dots, N$  do  
2:    $x_i \leftarrow \arg \min_{x \in \mathbb{R}^d} \{f(x) : x \in x_0 + \text{span}\{\nabla f(x_0), \dots, \nabla f(x_{i-1})\}\}$   
3:   Choose  $\nabla f(x_i) \in \partial f(x_i)$  such that  $\langle \nabla f(x_i), \nabla f(x_j) \rangle = 0, \forall j < i$   
4: end for
```

Using their PEP framework, Drori and Taylor establish upper bounds on the worst-case convergence of (GFM). They further introduce a procedure to construct fixed-step first-order methods that achieve the same worst-case convergence as (GFM). This procedure leverages the dual formulation of PEP, which provides a proof for the convergence bounds (see Remark 2.17) and then ensures that the proof remains valid for the constructed fixed-step method (see [32, Section 4]). The core idea behind this method is that (GFM) achieves the best possible convergence on functions structurally similar to the "worst function in the world," introduced by Nesterov in [71] to derive complexity lower bounds. Fixed-step first-order methods constructed using this procedure may therefore be optimal, as they exhibit the same worst-case behavior as (GFM). Drori and Taylor further demonstrate the effectiveness of their approach by showing, for instance, that (GFM) attains the best possible convergence bound for smooth convex functions and recover the optimal fixed-step algorithm for this class, (OGM).

The first two methods for designing fixed-step first-order algorithms presented above (SDP relaxations and the (GFM)-based method) have two main drawbacks. First, the optimality of the algorithm designed is not guaranteed and typically requires a separate proof. Second, these methods are problem-specific, meaning they may not apply successfully to all problem classes where a convex PEP can be formulated. For example, the SDP relaxation method fails in designing an optimized memoryless gradient descent algorithm for L -smooth convex optimization, i.e., optimizing the step sizes α_i in the following algorithm (see [48]):

Algorithm 6 Memoryless Gradient Descent (MGD)

```
1: Input: function  $f, x_0 \in \mathbb{R}^d$ .  
2: for  $i = 0$  to  $N - 1$  do  
3:    $x_{i+1} = x_i - \frac{\alpha_i}{L} \nabla f(x_i)$   
4: end for
```

Remark 2.23. Algorithm (MGD) is called memoryless because, unlike (OGM) and (FGM), which use past gradients at each update to achieve an accelerated convergence rate, (MGD) only utilizes the most recent gradient $\nabla f(x_i)$ for the update at each iteration. Memoryless fixed-step first-order algorithms are fixed-step algorithms of the form (2.8) with $\alpha_{i,k} = 0$ for all $k \neq i - 1$, and $\alpha_{i,i-1} = \frac{\alpha_{i-1}}{L}$. Note that (GM) is a special case of (MGD), where the step sizes are constant, $\alpha_i = \alpha$.

In [48], Das Gupta et al. address these two issues by introducing BNB-PEP, a unified methodology for designing optimal fixed-step first-order algorithms. They solve the nonconvex design problem (opt-PEP) to global optimality using a specialized spatial Branch-and-Bound method. The main drawback of this approach is the high computational cost of the Branch-and-Bound algorithm, which limits the number of steps N for which the step sizes $\{\alpha_i\}_{i \leq N}$ can be optimized. Despite this limitation, Das Gupta et al. successfully apply BNB-PEP to various algorithm design problems, obtaining meaningful results. Among their examples, they address the problem of designing an optimal (MGD) algorithm, deriving optimized numerical step sizes for N up to 50 (provably optimal for N up to 25). Their results suggest a convergence rate of $\mathcal{O}(1/N^{1.178})$, providing the first numerical evidence that an accelerated convergence rate (faster than $\mathcal{O}(1/N)$) may be achievable without memory (see Remark 2.23). The step-size schedule designed by Das Gupta et al. alternates between short and unusually long step sizes, violating the descent condition $0 < \alpha_i \leq \frac{2}{L}$. They further establish that acceleration is impossible for memoryless algorithms on smooth convex functions without such long steps (see [48, Section 6.1.1]).

Building on these insights, Grimmer [44] introduced the first numerical step-size schedule with provable acceleration guarantees. Shortly thereafter, Altschuler and Parrilo [7] proposed the *silver step-sizes*, the first analytically derived step-size schedule that guarantees acceleration for memoryless fixed-step first-order algorithms applied to smooth strongly convex functions. They later extended the silver step-size schedule to smooth (non-strongly) convex functions [8]. In this setting, the silver step sizes are defined recursively. For any $n \in \mathbb{N}$ such that $n = 2^k - 1$, the silver step-size schedule $\tilde{\alpha}_{2n+1} = [\alpha_0, \dots, \alpha_{2n}]$ of length $2n + 1$ is given by:

$$\tilde{\alpha}_{2n+1} = [\tilde{\alpha}_n, 1 + \rho^{k-1}, \tilde{\alpha}_n], \quad (2.14)$$

where $\rho = 1 + \sqrt{2}$ is the silver ratio, and $\tilde{\alpha}_1 = [\sqrt{2}]$. The convergence

of Algorithm (MGD) with the silver step-size schedule is stated in the following theorem:

Theorem 2.24. *For any function $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ and integer $N = 2^k - 1$, after N steps of (MGD) with the silver step-size schedule, the following holds:*

$$f(x_N) - f_* \leq r_k L \|x_0 - x_*\|^2,$$

$$\text{where } r_k := \frac{1}{1 + \sqrt{4\rho^{2^k} - 3}} \leq \frac{1}{2\rho^{\log_2 N}} = \frac{1}{2N^{\log_2 \rho}} \approx \frac{1}{2N^{1.2716}}.$$

Inspired by the silver step sizes, recent concurrent works [45, 108] provide a theoretical framework for designing optimized (MGD) algorithms for smooth convex functions. This approach involves composing step-size schedules that satisfy specific properties with appropriate composition operations. Grimmer et al. [45] show that the silver step sizes can be derived within this framework and then construct highly optimized schedules, conjectured to be optimal, for any number of steps N (not limited to $N = 2^k - 1$ as required by the silver step sizes).

2.6 Conclusion

In this first chapter, we illustrated the utility of the Performance Estimation Problem (PEP) framework by reviewing fundamental results on the worst-case analysis and design of fixed-step first-order methods in the smooth convex and smooth strongly convex settings. We first explained how, using the necessary and sufficient interpolation conditions (2.5), it is possible to derive a convex SDP formulation for the PEPs of fixed-step first-order algorithms. As a use case, we then reviewed existing results and conjectures on the exact worst-case behavior of the classical (GM) algorithm. Additionally, we examined the algorithm design process facilitated by PEP and some of the optimized fixed-step first-order algorithms derived from it. In particular, we presented (OGM) an optimal first-order algorithm for smooth convex functions and (ITEM) an optimal algorithm for smooth strongly convex functions, along with their performance analysis. Furthermore, we reviewed recent optimized memoryless algorithms (MGD), whose worst-case analyses demonstrate that Nesterov's acceleration is not necessary to achieve an accelerated convergence rate (i.e., faster than $\mathcal{O}(1/N)$).

In the remainder of this thesis, we extend the worst-case analysis and optimized algorithm design using PEP to a broader range of function classes and algorithms.

3

Performance Estimation of First-Order Block Coordinate-Wise Algorithms

3.1 Introduction

In this second chapter, we extend the Performance Estimation methodology to block coordinate descent algorithms (BCD), which successively update a single block of coordinates at a time. BCD algorithms constitute a widely used class of methods in modern optimization. Indeed, with the widespread availability of data, the scale of modern large-scale optimization problems is constantly increasing, making standard full-gradient optimization methods computationally expensive. Fortunately, many of these problems exhibit a block coordinate structure, making them well-suited for BCD algorithms. Among block coordinate-wise algorithms, an important subclass is block coordinate descent algorithms, which require access to only a subset of gradient coordinates at each iteration. These methods can generally be classified into three main categories based on how the blocks of coordinates are selected and updated [103]:

1. **Gauss-Southwell methods**, which greedily select the coordinates that lead to the largest improvement.

2. **Randomized methods**, which select coordinates according to a probability distribution.
3. **Cyclic methods**, which update the coordinates in a predefined cyclic order.

Although greedy methods can perform well, their update rule typically requires access to full gradients to identify which block of coordinates has the largest gradient norm and will lead to the most significant improvement. As a result, randomized and cyclic methods have been more extensively studied and utilized.

The worst-case convergence analysis of random coordinate descent methods has proven to be easier than that of their deterministic counterparts. Sampling coordinates with replacement from a suitable probability distribution ensures that the expectation of each coordinate step corresponds to the full gradient, making the analysis largely similar to that of full gradient descent. Consequently, numerous random coordinate descent algorithms with theoretical guarantees have been proposed for convex optimization problems, including accelerated and proximal variants for various probability distributions [72, 62, 36, 28, 6, 50, 75]. However, their performance guarantees hold only in expectation or with high probability, and the sampling technique can be computationally costly specially in high-dimension settings.

In practice, cyclic block coordinate descent is often preferred over random or greedy variants due to its simplicity and computational efficiency. It cycles through coordinates in a fixed order, avoiding the overhead of random sampling or the computational cost of selecting the most promising block at each step, as in greedy methods. It also tends to perform reliably well across a wide range of problems [49].

This motivates the study of cyclic block coordinate methods, which, although simpler to implement, are significantly harder to analyze. The primary challenge lies in establishing a connection between the block of coordinates updated at each step and the full gradient. Some convergence results are already known for cyclic coordinate descent, but they are often obtained under restrictive assumptions, such as the isotonicity of the gradient [88], or with worst-case bounds that are quite conservative and under initial conditions that are not entirely standard for unconstrained

convex optimization [14, 52]. Stronger convergence results exist for some specific cases, such as for quadratic and strongly convex optimization problems [93, 60, 104, 49, 41].

We consider the unconstrained optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x),$$

where f belongs to the class $\mathcal{F}_{0,+\infty}(\mathbb{R}^d)$, is differentiable, and satisfies specific block coordinate-wise smoothness properties. To formalize this, we partition the space $\mathbb{R}^d$ into p subspaces:

$$\mathbb{R}^d = \mathbb{R}^{d_1} \times \dots \times \mathbb{R}^{d_p}.$$

We introduce the corresponding selection matrices $U_\ell \in \mathbb{R}^{d \times d_\ell}$, such that:

$$(U_1, \dots, U_p) = \mathcal{I}_d.$$

For any $x \in \mathbb{R}^d$, we can express x in terms of its block components as:

$$x = (x^{(1)}, \dots, x^{(p)})^T, \quad \text{where} \quad x^{(\ell)} = U_\ell^\top x \in \mathbb{R}^{d_\ell}, \quad \forall \ell \in \{1, \dots, p\}. \quad (3.1)$$

Thus, x can be rewritten as:

$$x = \sum_{\ell=1}^p U_\ell x^{(\ell)}.$$

Definition 3.1. The partial gradients of f at x are defined as:

$$\nabla^{(\ell)} f(x) \triangleq U_\ell^\top \nabla f(x), \quad \forall \ell \in \{1, \dots, p\}.$$

We now define the class of coordinate-wise smooth convex functions, denoted $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$:

Definition 3.2 (Functional class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$). Given a vector of nonnegative constants $\mathbf{L} = (L_1, \dots, L_p)$ and a function f , we say that f belongs to $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ if and only if:

1. f belongs to $\mathcal{F}_{0,+\infty}(\mathbb{R}^d)$ and is differentiable.
2. For every block of coordinates, f satisfies the block coordinate-wise

smoothness condition:

$$\|\nabla^{(\ell)} f(x + U_\ell h^{(\ell)}) - \nabla^{(\ell)} f(x)\| \leq L_\ell \|h^{(\ell)}\|, \quad \forall x \in \mathbb{R}^d, \forall h^{(\ell)} \in \mathbb{R}^{d_\ell}.$$

For the analysis of first-order algorithms over the functional class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, we define the following two norms:

Definition 3.3. Given a vector $\mathbf{L} = (L_1, \dots, L_p)$ of nonnegative constants, for any $x, g \in \mathbb{R}^d$, we define the following weighted primal and dual norms:

$$\|x\|_{\mathbf{L}}^2 = \sum_{\ell=1}^p L_\ell \|x^{(\ell)}\|^2, \quad \text{and} \quad \|g\|_{\mathbf{L}}^{*2} = \sum_{\ell=1}^p \frac{1}{L_\ell} \|g^{(\ell)}\|^2. \quad (3.2)$$

Note that the function $\frac{\|\cdot\|_{\mathbf{L}}^{*2}}{2}$ is the Fenchel conjugate of $\frac{\|\cdot\|_{\mathbf{L}}^2}{2}$.

As this chapter focuses on the analysis of first-order block coordinate-wise algorithms, we define the partial first-order oracle for the block of coordinates $\ell \in \{1, \dots, p\}$, denoted by $\mathcal{O}_\ell(f, x)$, as:

$$\mathcal{O}_\ell(f, x) = \{f(x), \nabla^{(\ell)} f(x)\}.$$

Let $t : \{0, \dots, N-1\} \rightarrow \{1, \dots, p\}$ be a block coordinate selection strategy that associates each iteration $i \in \{1, \dots, N\}$ with a block of coordinates $t(i)$, such that the oracle $\mathcal{O}_{t(i)}(f, x)$ is used for the update at step i . A first-order block coordinate-wise algorithm $\mathcal{M}^{\text{coord}}$ can then be conceptually expressed as follows:

$$\begin{aligned} x_1 &= \mathcal{M}_1^{\text{coord}} \left(x_0, \mathcal{O}_{t(0)}(f, x_0) \right), \\ x_2 &= \mathcal{M}_2^{\text{coord}} \left(x_0, \mathcal{O}_{t(0)}(f, x_0), \mathcal{O}_{t(1)}(f, x_1) \right), \\ &\vdots \\ x_N &= \mathcal{M}_N^{\text{coord}} \left(x_0, \mathcal{O}_{t(0)}(f, x_0), \dots, \mathcal{O}_{t(N-1)}(f, x_{N-1}) \right). \end{aligned} \quad (3.3)$$

where $\mathcal{M}_i^{\text{coord}}$ represents the operation that generates the iterate x_i from the initial iterate x_0 and the past partial first-order information on the function f . We furthermore assume that the mappings $\mathcal{M}_i^{\text{coord}}$ are such that x_i and x_{i-1} differ only on their $t(i-1)$ block of coordinates. As we explained previously in Chapter 2, in the case of standard first-order methods over

the function class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$, the Performance Estimation Problem for the first-order BCD algorithm $\mathcal{M}^{\text{coord}}$ over the functional class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ can be expressed as the following infinite-dimensional problem:

$$\begin{aligned}
w^{\text{coord}}(R, \mathbf{L}, \mathcal{M}^{\text{coord}}, N, \mathcal{I}, \mathcal{P}) &= \sup_{f, x_0, \dots, x_N, x_*} \mathcal{P}(f, x_0, \dots, x_N, x_*), \\
\text{such that } f &\in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d), \\
x_* &\text{ is optimal for } f, \\
x_1, \dots, x_N &\text{ generated from } x_0 \text{ using (3.3)} \\
\mathcal{I}(f, x_0, \dots, x_N, x_*) &\leq R.
\end{aligned}
\tag{i-PEP-coord}$$

where $\mathcal{P}$ is a performance criterion, for instance

$$f(x_N) - f(x_*),$$

and $\mathcal{I}(f, x_0, \dots, x_N, x_*) \leq R$ is an initial condition that ensures that the worst-case of the algorithm is bounded, for instance

$$\|x_0 - x_*\| \leq R.$$

Following the PEP methodology summarized in Remark (1.2) and as explained in the previous chapter for first-order methods over the classes $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$, Problem (i-PEP-coord) can be simplified into a finite-dimensional problem using the notion of interpolability of a finite set by a functional class. The $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolability of a finite set is defined as follows:

Definition 3.4 ($\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolability). Let I be an index set, and consider the set of triples $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$, where $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. The set $\mathcal{S}$ is $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolable if and only if there exists a function $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ such that $g_i = \nabla f(x_i)$ and $f(x_i) = f_i, \forall i \in I$.

Since Problem (i-PEP-coord) only involves first-order information of the functional variable $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ at the iterates $x_i, i \in \{0, 1, \dots, N\}$ of the algorithm $\mathcal{M}^{\text{coord}}$ and at a minimizer x_* of f , $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolability allows us to replace the functional variable f with a finite $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolable set of decision variables:

$$\mathcal{S}_N = \{x_i, g_i, f_i\}_{i \in I} \in (\mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R})^{N+2},$$

where $I = \{0, 1, \dots, N, *\}$, such that Problem (i-PEP-coord) takes the finite-dimensional form

$$\begin{aligned}
 w^{\text{coord}}(R, \mathbf{L}, \mathcal{M}^{\text{coord}}, N, \mathcal{I}, \mathcal{P}) = & \sup_{\mathcal{S}_N = \{x_i, g_i, f_i\}_{i \in I}} \mathcal{P}(\mathcal{S}_N), \\
 & \text{such that } \mathcal{S}_N \text{ is } \mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)\text{-interpolable,} \\
 & x_1, \dots, x_N \text{ generated from } x_0 \text{ using (3.3),} \\
 & g_* = 0, \\
 & \mathcal{I}(\mathcal{S}_N) \leq R.
 \end{aligned}
 \tag{f-PEP-coord}$$

Contributions. In the remainder of this chapter, we begin in Section 3.2 by reviewing useful characterizations of the function class $\mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ and compare it, in terms of inclusion, with the function classes $\mathcal{F}_{0, L_{\min}}(\mathbb{R}^d)$ and $\mathcal{F}_{0, L_{\text{sum}}}(\mathbb{R}^d)$, where $L_{\min} = \min_{\ell \in \{1, \dots, p\}} L_\ell$ and $L_{\text{sum}} = \sum_{\ell=1}^p L_\ell$. This comparison allows us to derive lower and upper bounds on the convergence of BCD algorithms over the class $\mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$.

In Section 3.3, we extend the PEP framework to BCD algorithms over the functional class $\mathcal{F}_{0, L}(\mathbb{R}^d)$ and demonstrate how to obtain a convex reformulation of the PEP for the subclass of deterministic fixed-step first-order block coordinate-wise methods. We also briefly discuss how this framework can be extended to their random counterparts.

In Section 3.4, we leverage this PEP framework to investigate the worst-case behavior of several BCD algorithms, including block coordinate descent, alternating minimization, and a cyclic version of the random accelerated block coordinate descent algorithm from [36]. For the block coordinate descent algorithm, our bounds improve upon the best known bound from [14] by approximately a multiplicative factor of 10. For the alternating minimization algorithm, our bounds suggest that the bound from [14] is asymptotically tight. Additionally, we provide numerical worst-case convergence analyses for these algorithms under more natural and less restrictive assumptions than those typically made in the analysis of cyclic block coordinate-wise algorithms [14, 72]. For the cyclic version of the random accelerated algorithm presented in [36], we establish a lower bound on the worst-case convergence of the algorithm, suggesting that acceleration schemes for random block coordinate-wise algorithms do not necessarily generalize to deterministic

cyclic variants.

Note that the above results do not explicitly exploit the coordinate smoothness assumption, as they optimize over $\mathcal{F}_{0,L}(\mathbb{R}^d)$ not over $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. To confirm these numerical observations and obtain tighter bounds for the convergence of fixed-step first-order BCD algorithms over the functional class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, we present in Section 3.5 necessary interpolation conditions for the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ and derive a convex PEP framework for fixed-step first-order BCD algorithms directly over this class. We also describe how to adapt this framework to derive PEP-based optimized descent lemmas for deterministic algorithms and how to handle random algorithms.

In Section 3.6, we formally establish several results on the worst-case convergence of cyclic block coordinate descent for coordinate-wise smooth convex functions, including:

- An invariance property of the worst-case with respect to the norm $\|\cdot\|_L$ introduced earlier.
- A lower bound on the worst-case performance, which equals the number of blocks multiplied by the worst-case performance of full gradient descent on smooth convex functions.
- An improved descent lemma.
- An upper bound on the residual gradient norm for two-block coordinate descent.

Finally, in Section 3.7, we provide tighter numerical results on the worst-case convergence of the three BCD algorithms under consideration, which further confirm the observations made in Section 3.4. Additionally, we numerically analyze the dependence of our bounds on the number of coordinate blocks, as well as the evolution of the optimal step size for cyclic block coordinate descent with respect to the number of blocks and cycles.

Summary of the main analytically established Results. Among the contributions presented above, the following results are analytically established:

- We provide a new proof for a necessary and sufficient characterization for the class of coordinate-wise smooth convex functions $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ (characterization 3, Theorem 3.7).

- We use this characterization as interpolation conditions for this class of functions and prove that these conditions are sufficient for sets of cardinality at most 2, but not sufficient in general (Theorems 3.20 and 3.21).
- We prove a lower bound on the worst-case convergence rate of p -block cyclic coordinate descent over the class of coordinate-wise smooth functions which is equal to the number of blocks p times the worst-case convergence rate of full gradient descent on smooth convex functions (Theorem 3.27).
- We provide a new descent lemma for 2-block coordinate descent and use it to derive an upper bound on the convergence of the squared residual gradient norm (Lemma 3.30, Theorem 3.32).

3.2 Class of Coordinate-wise Smooth Functions

In this section, we present necessary and sufficient characterizations for the class of coordinate-wise smooth convex functions, which are essential for analyzing Block Coordinate Descent (BCD) algorithms using Performance Estimation. We begin with a valid quadratic upper bound for this class of functions.

Lemma 3.5. [72, Section 2] *Let p be the number of coordinate blocks, $\mathbf{L} = (L_1, \dots, L_p)$ a vector of nonnegative constants, and $f : \mathbb{R}^d \mapsto \mathbb{R}$ a differentiable function. If $f \in \mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, then:*

1. *f satisfies the following inequality:*

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle, \quad \forall x_1, x_2 \in \mathbb{R}^d.$$

2. *For all $\ell \in \{1, \dots, p\}$, $x \in \mathbb{R}^d$, and $h^{(\ell)} \in \mathbb{R}^{d_\ell}$, f satisfies the quadratic upper bound:*

$$f(x + U_\ell h^{(\ell)}) \leq f(x) + \langle \nabla^{(\ell)} f(x), h^{(\ell)} \rangle + \frac{L_\ell}{2} \|h^{(\ell)}\|^2. \quad (3.4)$$

We now provide a valid quadratic lower bound for coordinate-wise smooth convex functions.

Lemma 3.6. *Let p be the number of blocks of coordinates, $\mathbf{L} = (L_1, \dots, L_p)$ a vector of nonnegative constants, and $f : \mathbb{R}^d \mapsto \mathbb{R}$ a differentiable function. If*

$f \in \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, then for all $\ell \in \{1, \dots, p\}$ and for all $x_1, x_2 \in \mathbb{R}^d$, f satisfies the quadratic lower bound:

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_2)\|^2. \quad (3.5)$$

Proof. The following proof is inspired from the proof of [96, Theorem 2.34] where a similar result for L -smooth convex functions is derived. We define the subspace $S_{\neq \ell} = \{x \in \mathbb{R}^d, U_j^\top x = 0, \forall j \in 1, \dots, p, j \neq \ell\}$ and the extended function p_ℓ as follows:

$$p_\ell(x) = \frac{\|U_\ell^\top x\|^2}{2} + \mathbb{I}_{S_{\neq \ell}}(x),$$

with

$$\mathbb{I}_{S_{\neq \ell}}(x) = \begin{cases} 0 & \text{if } x \in S_{\neq \ell} \\ +\infty & \text{otherwise} \end{cases}$$

Since f belongs to $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ it satisfies condition (3.4). We claim that we can rewrite this condition as follows:

$$f(x + \Delta x) \leq f(x) + \langle \nabla f(x), \Delta x \rangle + L_\ell p_\ell(\Delta x) \quad \forall x \in \mathbb{R}^d, \forall \Delta x \in \mathbb{R}^d, \quad (3.6)$$

which now involves the full gradient of f as opposed to the partial gradient $\nabla^{(\ell)} f$, and must be satisfied for all $\Delta x \in \mathbb{R}^d$ as opposed to the $U_\ell h^{(\ell)}$ that are only nonzero along the block ℓ . Indeed, if Δx is only nonzero along the coordinates of the ℓ^{th} block, then $p_\ell(\Delta x) = \frac{\|U_\ell^\top \Delta x\|^2}{2}$, $\langle \nabla f(x), \Delta x \rangle = \langle \nabla^{(\ell)} f(x), U_\ell^\top \Delta x \rangle$ and we recover (3.4). For all other values of Δx , $p_\ell(\Delta x)$ is infinite, and (3.6) is trivially satisfied. For any fixed $x \in \mathbb{R}^d$, let us now define

$$h_x(\Delta x) = f(x + \Delta x) - f(x) - \langle \nabla f(x), \Delta x \rangle.$$

Condition (3.6) becomes then

$$h_x(\Delta x) \leq L_\ell p_\ell(\Delta x), \quad \forall x, \Delta x \in \mathbb{R}^d.$$

Note that both h_x and p_ℓ are convex. As the Fenchel conjugate (with respect to Δx) reverses the order between convex functions, this is equivalent to:

$$h_x^*(\Delta g) \geq (L_\ell p_\ell)^*(\Delta g), \quad \forall \Delta g \in \mathbb{R}^d. \quad (3.7)$$

First, we compute the Fenchel conjugate $(L_\ell p_\ell)^*(\Delta g)$. For any $\Delta x \in \mathbb{R}^d$ such that $\Delta x \notin S_{\neq \ell}$, we clearly have:

$$\langle \Delta g, \Delta x \rangle - L_\ell p_\ell(\Delta x) = -\infty$$

For the case where $\Delta x \in S_{\neq \ell}$, we have that:

$$\begin{aligned} \forall \Delta x \in S_{\neq \ell}, \Delta g \in \mathbb{R}^d, T(\Delta x, \Delta g) &= \langle \Delta g, \Delta x \rangle - L_\ell p_\ell(\Delta g) \\ &= \Delta g^T U_\ell U_\ell^T \Delta x - \frac{L_\ell}{2} \Delta x^T U_\ell U_\ell^T \Delta x \\ &= \Delta g^T M_\ell \Delta x - \frac{L_\ell}{2} \Delta x^T M_\ell \Delta x \end{aligned}$$

with $M_\ell = U_\ell U_\ell^T$ is an orthogonal projection matrix i.e $M_\ell = M_\ell^T = M_\ell^2$. Computing the derivative of T with respect to Δx and setting it to zero we have that the maximum, Δx^* verifies:

$$\Delta x^* = \frac{1}{L_\ell} M_\ell \Delta g \in S_{\neq \ell}$$

Using this we get that:

$$\Delta g^T M_\ell \Delta x^* = \frac{1}{L_\ell} \Delta g^T M_\ell \Delta g,$$

and:

$$\frac{L_\ell}{2} \Delta x^{*T} M_\ell \Delta x^* = \frac{L_\ell}{2} \Delta x^{*T} M_\ell^2 \Delta x^* = \frac{L_\ell}{2} (\Delta x^{*T} M_\ell^T) (M_\ell \Delta x^*) = \frac{1}{2L_\ell} \Delta g^T M_\ell \Delta g.$$

Putting everything together we have that:

$$\begin{aligned} \forall \Delta g \in \mathbb{R}^d, (L_\ell p_\ell)^*(\Delta g) &= \sup_{\Delta x \in \mathbb{R}^d} \langle \Delta g, \Delta x \rangle - L_\ell p_\ell(\Delta x) \\ &= \sup_{\Delta x \in S_{\neq \ell}} \langle \Delta g, \Delta x \rangle - L_\ell p_\ell(\Delta x) \\ &= \frac{1}{2L_\ell} \Delta g^T M_\ell \Delta g = \frac{1}{2L_\ell} \|U_\ell^T \Delta g\|^2. \end{aligned}$$

Let us now compute an expression of the Fenchel conjugate of h_x with

respect to Fenchel conjugate f^* of f :

$$\begin{aligned}
h_x^*(\Delta g) &= \sup_{\Delta x \in \mathbb{R}^d} \langle \Delta g, \Delta x \rangle - f(x + \Delta x) + f(x) + \langle \nabla f(x), \Delta x \rangle \\
&= f(x) + \sup_{\Delta x \in \mathbb{R}^d} \langle \nabla f(x) + \Delta g, \Delta x \rangle - f(x + \Delta x) \\
&= f(x) + \sup_{x' \in \mathbb{R}^d} \langle \nabla f(x) + \Delta g, x' - x \rangle - f(x') \quad \text{with } x' = x + \Delta x \\
&= f(x) - \langle \nabla f(x) + \Delta g, x \rangle + \sup_{x' \in \mathbb{R}^d} \langle \nabla f(x) + \Delta g, x' \rangle - f(x') \\
&= f(x) - \langle \nabla f(x) + \Delta g, x \rangle + f^*(\nabla f(x) + \Delta g).
\end{aligned}$$

By Young's equality (see for instance [96, Theorem 2.21]), we have that for any function $f \in \mathcal{F}_{0,+\infty}(\mathbb{R}^d)$ and $x, s \in \mathbb{R}^d$ such that $s \in \partial f(x)$:

$$f(x) + f^*(s) = \langle x, s \rangle.$$

Applying this to our case, we have that

$$f^*(\nabla f(x)) = f(x) - \langle \nabla f(x), x \rangle$$

which gives us

$$h_x^*(\Delta g) = f^*(\nabla f(x) + \Delta g) - f^*(\nabla f(x)) - \langle x, \Delta g \rangle.$$

Finally, we obtain from inequality (3.7) that condition (3.6) is equivalent to

$$\forall x \in \mathbb{R}^d, \forall \Delta g \in \mathbb{R}^d, f^*(\nabla f(x) + \Delta g) \geq f^*(\nabla f(x)) + \langle x, \Delta g \rangle + \frac{1}{2L_\ell} \|U_\ell^T \Delta g\|^2.$$

In particular, for any $x_1, x_2 \in \mathbb{R}^d$, choosing $x = x_2$ and $\Delta g = \nabla f(x_1) - \nabla f(x_2)$ in the previous inequality, we have that:

$$\begin{aligned}
f^*(\nabla f(x_1)) &\geq f^*(\nabla f(x_2)) + \langle x_2, \nabla f(x_1) - \nabla f(x_2) \rangle \\
&\quad + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_2)\|^2, \quad \forall x_1, x_2 \in \mathbb{R}^d.
\end{aligned}$$

By Young's equality again, we have that:

$$f^*(\nabla f(x_1)) = \langle x_1, \nabla f(x_1) \rangle - f(x_1),$$

$$f^*(\nabla f(x_2)) = \langle x_2, \nabla f(x_2) \rangle - f(x_2).$$

Injecting this two inequalities in the previous inequality, we find that:

$$\begin{aligned} f(x_2) &\geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle \\ &\quad + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_2)\|^2, \quad \forall x_1, x_2 \in \mathbb{R}^d. \end{aligned}$$

which concludes the proof. $\square$

We now prove that the characterizations of coordinate-wise smooth convex functions given in Lemmas 3.5 and 3.6 are necessary and sufficient conditions.

Theorem 3.7. *Let p be the number of blocks of coordinates, $\mathbf{L} = (L_1, \dots, L_p)$ a vector of nonnegative constants, and $f : \mathbb{R}^d \mapsto \mathbb{R}$ a differentiable function. The following statements are equivalent:*

1. $f \in \mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$.
2. For all $x_1, x_2 \in \mathbb{R}^d$,

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle,$$

and for all $\ell \in \{1, \dots, p\}$, $x \in \mathbb{R}^d$, and $h^{(\ell)} \in \mathbb{R}^{d_\ell}$,

$$f(x + U_\ell h^{(\ell)}) \leq f(x) + \langle \nabla^{(\ell)} f(x), h^{(\ell)} \rangle + \frac{L_\ell}{2} \|h^{(\ell)}\|^2.$$

3. For all $\ell \in \{1, \dots, p\}$ and $x_1, x_2 \in \mathbb{R}^d$,

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_2)\|^2.$$

Note that in 2. the second inequality must hold between two points differing only on a single block while in 3. it must hold for any pair $x_1, x_2 \in \mathbb{R}^d$.

Proof. The implication (1) $\implies$ (2) follows from Lemma 3.5, and (2) $\implies$ (3) is given by Lemma 3.6. To complete the proof, we show (3) $\implies$ (1). Assume f satisfies condition (3). Then, for all $x_1, x_2 \in \mathbb{R}^d$,

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle.$$

Now, applying condition (3) with $x_2 = x + U_\ell h^{(\ell)}$ and $x_1 = x$, for any $x \in \mathbb{R}^d$ and $h^{(\ell)} \in \mathbb{R}^{d_\ell}$, we get:

$$f(x + U_\ell h^{(\ell)}) \geq f(x) + \langle \nabla f(x), U_\ell h^{(\ell)} \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x) - \nabla^{(\ell)} f(x + U_\ell h^{(\ell)})\|^2.$$

Reversing the roles of x and $x + U_\ell h^{(\ell)}$ yields:

$$\begin{aligned} f(x) &\geq f(x + U_\ell h^{(\ell)}) + \langle \nabla f(x + U_\ell h^{(\ell)}), -U_\ell h^{(\ell)} \rangle \\ &\quad + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x) - \nabla^{(\ell)} f(x + U_\ell h^{(\ell)})\|^2. \end{aligned}$$

Adding these two inequalities gives:

$$\frac{1}{L_\ell} \|\nabla^{(\ell)} f(x) - \nabla^{(\ell)} f(x + U_\ell h^{(\ell)})\|^2 \leq \langle \nabla f(x + U_\ell h^{(\ell)}) - \nabla f(x), U_\ell h^{(\ell)} \rangle.$$

By the Cauchy-Schwarz inequality, we have:

$$\langle \nabla f(x + U_\ell h^{(\ell)}) - \nabla f(x), U_\ell h^{(\ell)} \rangle \leq \|\nabla f(x + U_\ell h^{(\ell)}) - \nabla f(x)\| \|U_\ell h^{(\ell)}\|,$$

which implies:

$$\|\nabla^{(\ell)} f(x + U_\ell h^{(\ell)}) - \nabla^{(\ell)} f(x)\| \leq L_\ell \|U_\ell h^{(\ell)}\|.$$

Since $\|U_\ell h^{(\ell)}\| = \|h^{(\ell)}\|$, this proves $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. $\square$

Remark 3.8. Note that characterization 3 in the previous theorem, when satisfied for any two vectors in $\mathbb{R}^d$, imposes both the convexity and coordinate-wise smoothness constraints on the function simultaneously, whereas characterization 2 imposes convexity and coordinate-wise smoothness separately using two different inequalities. Previous work on PEP (see [96]) shows that combining convexity and smoothness constraints, when possible, into a single constraint yields better interpolation conditions and thus tighter convergence rates on the algorithm performance. In our case, numerical experiments confirm that using characterization 3 as interpolation conditions in the PEP yields the tightest convergence bounds. Furthermore, numerical experiments show that combining the two characterizations 2 and 3 in the PEP does not lead to any improvement in the obtained worst-case. Finally, we compare in terms of inclusions the class of smooth convex functions with the class of coordinate-wise smooth convex functions.

Theorem 3.9. Let p be the number of blocks of coordinates and $\mathbf{L} = (L_1, \dots, L_p)$ a vector of nonnegative constants. Then:

$$\mathcal{F}_{0, L_{\min}}(\mathbb{R}^d) \subseteq \mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d) \subseteq \mathcal{F}_{0, L_{\text{sum}}}(\mathbb{R}^d),$$

where $L_{\min} = \min_{i \in \{1, \dots, p\}} L_i$ and $L_{\text{sum}} = \sum_{i=1}^p L_i$.

Proof. If $f \in \mathcal{F}_{0, L_{\min}}(\mathbb{R}^d)$, then f is convex and satisfies:

$$\|\nabla f(x_2) - \nabla f(x_1)\| \leq L_{\min} \|x_2 - x_1\|, \quad \forall x_1, x_2 \in \mathbb{R}^d.$$

Since $\|\nabla f(x_2) - \nabla f(x_1)\| = \sqrt{\sum_{i=1}^p \|\nabla^{(\ell)} f(x_2) - \nabla^{(\ell)} f(x_1)\|^2}$, it follows that:

$$\|\nabla^{(\ell)} f(x_2) - \nabla^{(\ell)} f(x_1)\| \leq L_{\min} \|x_2 - x_1\| \leq L_{\ell} \|x_2 - x_1\|.$$

Since the previous inequality is valid for any $x_1, x_2 \in \mathbb{R}^d$, we can choose $x_2 = x + U_{\ell} h^{(\ell)}$ and $x_1 = x$ with any $x \in \mathbb{R}^d$ and $h^{(\ell)} \in \mathbb{R}^{d_{\ell}}$ which give us

$$\|\nabla^{(\ell)} f(x + U_{\ell} h^{(\ell)}) - \nabla^{(\ell)} f(x)\| \leq L_{\ell} \|h^{(\ell)}\|, \quad \forall x \in \mathbb{R}^d, \forall h^{(\ell)} \in \mathbb{R}^{d_{\ell}}.$$

and thus $f \in \mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, proving the first inclusion. Now assume that f belongs to $\mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. By Theorem 3.7, for any $x_1, x_2 \in \mathbb{R}^d$ and $\ell \in \{1, \dots, p\}$, it holds that

$$2L_{\ell}(f(x_2) - f(x_1) - \langle \nabla f(x_1), x_2 - x_1 \rangle) \geq \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_2)\|^2.$$

Summing these inequalities for $\ell \in 1, \dots, p$, we have

$$2L_{\text{sum}}(f(x_2) - f(x_1) - \langle \nabla f(x_1), x_2 - x_1 \rangle) \geq \sum_{\ell=1}^p \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_2)\|^2.$$

Since

$$\sum_{\ell=1}^p \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_2)\|^2 = \|\nabla f(x_1) - \nabla f(x_2)\|^2,$$

we find that for any $x_1, x_2 \in \mathbb{R}^d$

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{1}{2L_{\text{sum}}} \|\nabla f(x_1) - \nabla f(x_2)\|^2,$$

which proves that f belongs to $\mathcal{F}_{0, L_{\text{sum}}}(\mathbb{R}^d)$ and concludes the proof. $\square$

The above theorem implies that the worst-cases of any algorithm on the classes $\mathcal{F}_{0,L_{\min}}(\mathbb{R}^d)$ and $\mathcal{F}_{0,L_{\text{sum}}}(\mathbb{R}^d)$ provide respectively valid lower and upper bounds for its worst-case convergence on the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$.

3.3 Convex PEP for Fixed-Step BCD Algorithms on the Functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$

In this section, we extend the convex Performance Estimation Problem (PEP) framework for fixed-step first-order algorithms on smooth convex functions, originally developed in [97] and reviewed in Chapter 2, to fixed-step first-order block coordinate-wise algorithms. Note that we consider $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$ and not yet $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. Hence Theorem 3.9 will play some role (which will come later in Section 3.4).

Recall from Chapter 2, Section 2.3, that the Performance Estimation Problem for the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ admits a finite-dimensional formulation by defining the set of decision variables:

$$\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I} \in (\mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R})^{N+2},$$

where $I = \{0, \dots, N, *\}$. For all $i \in I$, the variable $g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ are constrained by the $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ -interpolability condition to be the gradient and function value at x_i for some function $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$. The variables x_i for $i \in \{0, \dots, N\}$ represent the first N iterates of the algorithm under analysis, and x_* is enforced to be an optimal point of f by the condition $g_* = 0$. Using this formulation, the PEP for the first-order BCD algorithm $\mathcal{M}^{\text{coord}}$ can be expressed as:

$$\sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} \mathcal{P}(\mathcal{S}_N),$$

subject to

$$f_i - f_j - \langle g_j, x_i - x_j \rangle \geq \frac{1}{2L} \|g_i - g_j\|^2, \quad \forall i, j \in I,$$

$x_1, \dots, x_N$ is generated from x_0 by the method $\mathcal{M}^{\text{coord}}$ with (3.3),

$$\{x_*, g_*, f_*\} = \{0, 0, 0\},$$

$$\mathcal{I}(\mathcal{S}_N) \leq R.$$

This formulation is a direct extension of Problem (d-PEP-Smooth) with $\mu = 0$, where the first-order algorithm $\mathcal{M}$ is replaced by the first-order BCD algorithm $\mathcal{M}^{\text{coord}}$. We now demonstrate that for fixed-step first-order BCD algorithms, the above problem can be reformulated in a convex form. Fixed-step BCD algorithms are defined as follows:

Definition 3.10. Given a number of blocks p , an integer N , and a block of coordinates selection strategy $t : \{0, \dots, N-1\} \rightarrow \{1, \dots, p\}$ that associates each step i of the algorithm with the index $t(i)$ of a block of coordinates, we say that $\mathcal{M}^{\text{coord}}$ is a fixed-step BCD algorithm if, for all $i \in \{1, \dots, N\}$, the iterate x_i can be expressed as:

$$x_i = x_0 - \sum_{k=0}^{i-1} \alpha_{i,k} U_{t(k)} \nabla^{(t(k))} f(x_k), \quad (3.8)$$

In the sequel, we denote by $\mathcal{M}_\alpha^{\text{coord}}$ the fixed-step BCD algorithm parametrized by the step sizes $\alpha_{i,k}$.

Remark 3.11. When the block coordinate selection strategy t is deterministic, for instance for the widely used strategy

$$t(i) = (i \bmod p) + 1,$$

which defines a cyclic order selection for the blocks, the BCD algorithm is called *deterministic*. Another possible choice is a random strategy, where the block of coordinates is sampled from a probability distribution. In this case, the BCD algorithm is called *random*.

Following a reasoning similar to the one used in Chapter 2, Section 2.3, we begin by extending the PEP framework to deterministic fixed-step first-order block coordinate-wise algorithms. Consider N iterations of the deterministic fixed-step BCD algorithm $\mathcal{M}_\alpha^{\text{coord}}$. Define the matrix $P^{(\ell)}$ by grouping the vectors $g_i^{(\ell)}$, corresponding to the ℓ -th block of coordinates of g_i , and the vector $x_0^{(\ell)}$, corresponding to the ℓ -th block of coordinates of the initial iterate x_0 :

$$P^{(\ell)} = [g_0^{(\ell)}, g_1^{(\ell)}, \dots, g_N^{(\ell)}, x_0^{(\ell)}].$$

The Gram matrix $G^{(\ell)}$ is then defined as $G^{(\ell)} = P^{(\ell)\top} P^{(\ell)} \in \mathbb{S}^{N+2}$. Now, let $h_i^{(\ell)} \in \mathbb{R}^{N+2}$ be defined component-wise as:

$$h_i^{(\ell)}(k) = \begin{cases} 1 & \text{if } k = N + 2, \\ -\alpha_{i,k-1} & \text{if } k \leq i \text{ and } t(k-1) = \ell, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $h_i^{(\ell)}(k)$ is the k -th element of the vector $h_i^{(\ell)}$. Then, by the definition of the BCD algorithm $\mathcal{M}_\alpha^{\text{coord}}$ in (3.8), the ℓ -th block of coordinates of the iterate x_i can be expressed as:

$$x_i^{(\ell)} = P^{(\ell)} h_i^{(\ell)}.$$

Defining $h_*^{(\ell)\top} = [0, \dots, 0] \in \mathbb{R}^{N+2}$, we obtain:

$$x_*^{(\ell)} = P^{(\ell)} h_*^{(\ell)} \quad \forall \ell \in \{1, \dots, p\}.$$

Similarly, let $u_i = e_{i+1} \in \mathbb{R}^{N+2}$, where $\{e_j\}_{j=1}^{N+2}$ is the canonical basis of $\mathbb{R}^{N+2}$, and set $u_* = 0$. Then, for all $i \in \{0, \dots, N\}$ and $\ell \in \{1, \dots, p\}$, the ℓ -th block of coordinates of the gradient g_i can be written as:

$$g_i^{(\ell)} = P^{(\ell)} u_i, \quad g_*^{(\ell)} = P^{(\ell)} u_*.$$

Given the previous definitions, for all $k, j \in \{0, \dots, N, *\}$ and $\ell \in \{1, \dots, p\}$, we obtain:

$$\begin{aligned} \langle x_i^{(\ell)}, x_j^{(\ell)} \rangle &= h_i^{(\ell)\top} G^{(\ell)} h_j^{(\ell)}, \\ \langle g_i^{(\ell)}, g_j^{(\ell)} \rangle &= u_i^\top G^{(\ell)} u_j, \\ \langle g_i^{(\ell)}, x_j^{(\ell)} \rangle &= u_i^\top G^{(\ell)} h_j^{(\ell)}. \end{aligned}$$

Recall that the interpolation conditions for the functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$ are given by:

$$f_i - f_j \geq \langle g_j, x_i - x_j \rangle + \frac{1}{2L} \|g_i - g_j\|^2, \quad \forall i, j \in I.$$

These conditions can be rewritten in block coordinate-wise form as:

$$f_i - f_j \geq \sum_{\ell=1}^p \langle g_j^{(\ell)}, x_i^{(\ell)} - x_j^{(\ell)} \rangle + \frac{1}{2L} \sum_{\ell=1}^p \|g_i^{(\ell)} - g_j^{(\ell)}\|^2, \quad \forall i, j \in I.$$

Rewriting this, we obtain:

$$f_i - f_j \geq \sum_{\ell=1}^p u_j^\top G^{(\ell)} (h_i^{(\ell)} - h_j^{(\ell)}) + \frac{1}{2L} \sum_{\ell=1}^p (u_i - u_j)^\top G^{(\ell)} (u_i - u_j), \quad \forall i, j \in I.$$

To express these conditions more compactly, we define the vector of function values $F^\top = [f_0, \dots, f_N]$ and the matrices:

$$2A_{i,j}^{(\ell)}(\alpha) = u_j(h_i^{(\ell)} - h_j^{(\ell)})^\top + (h_i^{(\ell)} - h_j^{(\ell)})u_j^\top + \frac{1}{L}(u_i - u_j)(u_i - u_j)^\top.$$

The interpolation conditions then become:

$$F^\top (u_j - u_i) + \sum_{\ell=1}^p \text{Tr} \left(A_{i,j}^{(\ell)}(\alpha) G^{(\ell)} \right) \leq 0.$$

For the initial condition $\mathcal{I}$, we consider conditions that can be expressed in the form:

$$\mathcal{I}(G^{(\ell)}) = \sum_{\ell=1}^p \text{Tr}(A_R^{(\ell)} G^{(\ell)}).$$

This formulation encompasses classical conditions, such as bounds on the distance $\|x_0 - x_*\|$. For the performance criterion $\mathcal{P}$, as explained in Chapter 2, Section 2.3, we consider a linear performance criterion of the form:

$$\mathcal{P}(F, G^{(\ell)}) = b^\top F + \sum_{\ell=1}^p \text{Tr}(C^{(\ell)} G^{(\ell)}).$$

Similarly to the case of standard fixed-step first-order algorithms, in the large-scale setting, when $d_\ell \geq N + 2$, $\forall \ell \in \{1, \dots, p\}$ (see Theorem 2.12 and the above discussion), the worst-case performance after N steps of the fixed-step first-order block coordinate descent algorithm $\mathcal{M}_\alpha^{\text{coord}}$ over the functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$ is given by the optimal solution of the following convex Performance Estimation Problem (PEP):

$$\begin{aligned}
w(R, L, \alpha, N, b, C^{(\ell)}) &= \sup_{G^{(\ell)}, F} b^\top F + \sum_{\ell=1}^p \text{Tr}(C^{(\ell)} G^{(\ell)}) \\
&\text{subject to:} \\
&F^\top (u_j - u_i) \\
&+ \sum_{\ell=1}^p \text{Tr} \left(A_{i,j}^{(\ell)}(\alpha) G^{(\ell)} \right) \leq 0, \quad (\text{coord-PEP-Smooth}) \\
&\forall i, j \in I, \\
&\sum_{\ell=1}^p \text{Tr}(A_R^{(\ell)} G^{(\ell)}) \leq R^2, \\
&G^{(\ell)} \succeq 0, \quad \forall \ell \in \{1, \dots, p\}.
\end{aligned}$$

Since Problem (coord-PEP-Smooth) is dimension-independent, it also provides the exact worst-case convergence of the fixed-step first-order BCD algorithm under analysis over the set $\mathcal{F}_{0,L} = \bigcup_{d \in \mathbb{N}} \mathcal{F}_{0,L}(\mathbb{R}^d)$ of all L -smooth convex functions.

Remark 3.12. Imposing the dimensions of the blocks in our PEP framework requires enforcing rank constraints on the matrices $G^{(\ell)}$, which are nonconvex. For instance, analyzing coordinate descent (where all blocks have dimension 1) would require imposing the following rank constraints: $\text{rank}(G^{(\ell)}) = 1, \forall \ell \in \{1, \dots, p\}$. Thus, we can only provide valid upper bounds on p -block BCD algorithms without being able to make any assumption on the dimension of each block.

Note that Problem (coord-PEP-Smooth) is similar to (SDP-PEP-Smooth). The key distinction is that, since we are analyzing block-coordinate-wise algorithms, the formulation requires block coordinate-wise computations. In the PEP framework, this results in defining multiple matrix variables $G^{(\ell)}$, each of size $(N+2) \times (N+2)$, corresponding to different coordinate blocks. Consequently, solving a PEP for fixed-step first-order BCD algorithms is computationally more expensive than solving a PEP for standard fixed-step first-order methods. This limits the number of blocks that can be considered in our PEP framework, particularly when solving the resulting SDP with standard interior-point solvers. Improvements in computational efficiency can potentially be achieved at the cost of losing precision, for instance by using first-order method solvers or by relaxing the SDP constraints (e.g.,

augmented Lagrangian methods or spectral relaxations).

We conclude this section by demonstrating how the PEP formulation developed for deterministic algorithms can be extended to handle random fixed-step first-order BCD methods. If p is the number of blocks and N the number of steps, then a random algorithm can generate up to p^N different sequences of block of coordinates. Each sequence corresponds to a deterministic algorithm, denoted by $\mathcal{M}_r^{\text{coord}}$ for $r \in \{1, \dots, p^N\}$, which can be represented and analyzed using PEP with a corresponding set $\mathcal{S}_N^r = \{(x_{i,r}, g_{i,r}, f_{i,r})\}_{i \in I}$ with $I = \{0, 1, \dots, N, *\}$, as described earlier in this section. To represent the average performance of the random algorithm under analysis, we need to impose that all the corresponding deterministic algorithms are applied on the same function. This is done by defining the set $\mathcal{S}_N = \bigcup_{r=1}^{p^N} \mathcal{S}_N^r$ and imposing the interpolation conditions on $\mathcal{S}_N$ to ensure that all the sets $\mathcal{S}_N^r$ are interpolable by the same function. Furthermore, we also need to impose the same initial iterate for all the algorithms and measure the performance with respect to the same optimal point. This is done in PEP by imposing the conditions $x_{0,r} = x_0$ and $\{x_{*,r}, g_{*,r}, f_{*,r}\} = \{0, 0, 0\}$ for all $r \in \{1, \dots, p^N\}$. We then choose the expectation of the performance of each deterministic algorithm $\mathcal{M}_r^{\text{coord}}$ as the performance criterion. Consequently, the PEP for a random algorithm can be conceptually formulated as:

$$\begin{aligned}
& \sup_{\mathcal{S}_N^r = \{(x_{i,r}, g_{i,r}, f_{i,r})\}_{i \in I}} \sum_{r=1}^{p^N} \mathbb{P}_r \mathcal{P}(\mathcal{S}_N^r), \\
& \text{such that } \mathcal{S}_N = \bigcup_{r=1}^{p^N} \mathcal{S}_N^r \\
& \mathcal{S}_N \text{ is } \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)\text{-interpolable,} \\
& x_{1,r}, \dots, x_{N,r} \text{ by method } \mathcal{M}_r^{\text{coord}} \quad \forall r \in \{1, \dots, p^N\}, \\
& x_{0,r} = x_0 \quad \forall r \in \{1, \dots, p^N\} \\
& \{x_{*,r}, g_{*,r}, f_{*,r}\} = \{0, 0, 0\}, \quad \forall r \in \{1, \dots, p^N\} \\
& \|x_0 - x_*\|_2 \leq R,
\end{aligned} \tag{3.9}$$

where $\mathbb{P}_r$ represents the probability that the sequence r of blocks occurs. For fixed-step first-order random BCD algorithms, this problem can be cast as a convex semidefinite program using a similar construction to the one presented above for fixed-step first-order deterministic algorithms. Indeed,

by defining the matrices

$$P^{(\ell)} = [g_0^{(\ell)}, g_{1,1}^{(\ell)}, \dots, g_{N,1}^{(\ell)}, \dots, g_{1,p^N}^{(\ell)}, g_{1,p^N}^{(\ell)}, \dots, g_{N,p^N}^{(\ell)}, x_0^{(\ell)}],$$

the Gram matrices $G^{(\ell)} = P^{(\ell)\top} P^{(\ell)} \in \mathbb{S}^{N+2}$ and the vector of function values $F^\top = [f_0, f_{1,1}, \dots, f_{N,1}, \dots, f_{1,p^N}, \dots, f_{N,p^N}]$, the interpolation conditions on the set $\mathcal{S}_N = \bigcup_{r=1}^{p^N} \mathcal{S}_N^r$ can be written in the form

$$\forall i, j \in \{0, 1, \dots, Np^N, *\}, F^\top(u_j - u_i) + \sum_{\ell=1}^p \text{Tr} \left(A_{i,j}^{(\ell)}(\alpha) G^{(\ell)} \right) \leq 0,$$

with suitable vectors u_i, u_j and matrices $A_{i,j}^{(\ell)}(\alpha)$, whose construction is very similar to the deterministic case presented previously. This leads to the following SDP:

$$w^{\text{rdn}}(R, L, \alpha, N, b_r, C_r^{(\ell)}) = \sup_{G^{(\ell)}, F} \sum_{r=1}^{p^N} \mathbb{P}_r \left(b_r^\top F + \sum_{\ell=1}^p \text{Tr}(C_r^{(\ell)} G^{(\ell)}) \right)$$

subject to:

$$F^\top(u_j - u_i)$$

$$+ \sum_{\ell=1}^p \text{Tr} \left(A_{i,j}^{(\ell)}(\alpha) G^{(\ell)} \right) \leq 0,$$

$$\forall i, j \in \{0, 1, \dots, Np^N, *\},$$

$$\sum_{\ell=1}^p \text{Tr}(A_R^{(\ell)} G^{(\ell)}) \leq R^2,$$

$$G^{(\ell)} \succeq 0, \quad \forall \ell \in \{1, \dots, p\}.$$

(PEP-rdn-Smooth)

Similarly to the deterministic case, if $d_\ell \geq N + 2$ for all $\ell \in \{1, \dots, p\}$, then the optimal solution of Problem (PEP-rdn-Smooth) provides the exact worst-case expectation of the performance of the analyzed random fixed-step first-order BCD algorithm over the functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$. Since Problem (PEP-rdn-Smooth) is also dimension-independent, it provides the exact worst-case expectation of the algorithm's performance over the set of all L -smooth convex functions, given by $\mathcal{F}_{0,L} = \bigcup_{d \in \mathbb{N}} \mathcal{F}_{0,L}(\mathbb{R}^d)$. However, the resulting SDP involves p matrices of size $(Np^N + 2) \times (Np^N + 2)$, making it computationally prohibitive for analyzing random algorithms when

the number of blocks and steps are large.

3.4 Numerical Experiments on the Convergence of Block Coordinate-wise Algorithms on the class $\mathcal{F}_{0,L}(\mathbb{R}^d)$.

We now leverage the PEP framework established in the previous section to analyze the convergence of three specific cyclic BCD algorithms on smooth convex functions. Given a vector of nonnegative smoothness constants $\mathbf{L}$, we analyze these algorithms over both the classes $\mathcal{F}_{0,L_{\min}}(\mathbb{R}^d)$ and $\mathcal{F}_{0,L_{\text{sum}}}(\mathbb{R}^d)$, which, thanks to Theorem 3.9, provide us respectively with a lower and an upper bound on the worst-case convergence of the algorithms over the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. As previously mentioned, cyclic algorithms form a subclass of deterministic BCD methods, where the blocks of coordinates to be updated are selected in a predefined cyclic order.

Let K be the number of cycles performed by a cyclic BCD algorithm and p the number of blocks of coordinates. Then, the total number of steps performed by the algorithm is given by $N = pK$. For the analysis of the considered algorithms, we choose the functional accuracy $f(x_N) - f(x_*)$ as the performance criterion $\mathcal{P}$. This criterion can be easily incorporated into the PEP framework developed in the previous section. Regarding the initial condition, a common assumption in the analysis of cyclic BCD algorithms, as adopted in [14, 72], is that the level set

$$S = \{x \in \mathbb{R}^d : f(x) \leq f(x_0)\}$$

is compact, which implies the following bounds (using the fact that the algorithm is decreasing the objective monotonically):

$$\|x_{pk} - x_*\| \leq R(x_0), \quad \forall k \in \{1, \dots, K\}, \quad (3.10)$$

where $R(x_0)$ is defined as:

$$R(x_0) = \max_{x_* \in X_*} \max_{x \in \mathbb{R}^d} \{\|x - x_*\| : f(x) \leq f(x_0)\}, \quad (3.11)$$

with X_* is the set of optimal points of the function. To define a setting compatible with our PEP framework while ensuring that the convergence theorems in [14] remain applicable and their bounds remain valid for comparison, we introduce the following Setting ALL.

Setting ALL. We assume that the iterates satisfy:

$$\min_{x_* \in X_*} \max_{k \in \{1, \dots, K\}} \|x_{pk} - x_*\| \leq R_a. \quad (3.12)$$

Note that the condition (3.12) in Setting ALL can be used as an initial condition in our PEP framework, as it is equivalent to imposing:

$$\|x_{pk} - x_*\| \leq R_a, \quad \forall k \in \{1, \dots, K\},$$

which can be expressed in matrix form as:

$$\sum_{\ell=1}^p \text{Tr}(A_R^{(\ell)} G^{(\ell)}) \leq R_a^2, \quad \forall k \in \{1, \dots, K\},$$

where $A_R^{(\ell)} = h_{pk}^{(\ell)} h_{pk}^{(\ell)\top}$.

Though theoretically convenient, Setting ALL is not very natural and may be difficult to verify in practice. In addition, Setting ALL can prove to be unusable for certain class of functions. Indeed, consider the family of smooth functions $f_\epsilon(x, y) = (x - y)^2 + \epsilon(x^2 + y^2)$ and the initial point $(x_0 = 1, y_0 = -1)$. Computing the gradients along each of the two blocks of coordinates gives

$$\begin{aligned} \forall x, y \in \mathbb{R}^d, \quad \nabla^{(x)} f(x, y) &= 2(1 + \epsilon)x - 2y, \\ \forall x, y \in \mathbb{R}^d, \quad \nabla^{(y)} f(x, y) &= 2(1 + \epsilon)y - 2x. \end{aligned}$$

This implies that f_ϵ is $2(1 + \epsilon)$ -smooth along each coordinate. Applying the 2-block cyclic coordinate descent algorithm with the step size $\frac{1}{2(1+\epsilon)}$, starting along x , we obtain that after one cycle of the algorithm:

$$\begin{aligned} x_{p(k+1)} &= \frac{y_{pk}}{(1 + \epsilon)}, \\ y_{p(k+1)} &= \frac{y_{pk}}{(1 + \epsilon)^2}. \end{aligned}$$

We have that $R(x_0) = \min_{x_* \in X_*} \max_{k \in \{1, \dots, K\}} \|x_{pk} - x_*\| = \|(x_2, y_2)\| = \frac{1}{(1+\epsilon)^2} \sqrt{1 + \frac{1}{(1+\epsilon)^2}}$, which tends to infinity as ϵ tends to zero. For ϵ small

enough $R(x_0)$ becomes very large and the bounds obtained in this setting are very conservative and do not give useful information about the performance of the algorithm. Therefore, we will also consider a more classical setting, albeit less frequently used in the context of cyclic block coordinate-wise algorithms:

Setting INIT. Given the starting point of the block coordinate algorithm x_0 and an optimal point of the function x_* , we have that

$$\|x_0 - x_*\|_L \leq R_i, \quad (3.13)$$

with $\|x_0 - x_*\|_L^2 = L\|x_0 - x_*\|^2$. The condition (3.13) in Setting INIT is also compatible with our PEP framework for BCD algorithms as using the notations of the previous section, it can be written under the form:

$$\sum_{\ell=1}^p \text{Tr}(A_R^{(\ell)} G^{(\ell)}) \leq R_i^2$$

with $A_R^{(\ell)} = L u_{N+1} u_{N+1}^\top$.

We now introduce the three cyclic BCD algorithms we will analyze using our PEP framework along with the corresponding tightest known upper bounds on their worst-case convergence rates. First, the cyclic block coordinate descent algorithm (CCD) performs a gradient step at each iteration with respect to a block of coordinates selected in a cyclic order.

Algorithm 7 Cyclic Coordinate Descent (CCD)

Input function f defined over $\mathbb{R}^d$ with p blocks, starting point $x_0 \in \mathbb{R}^d$, number of cycles K and step-sizes $\{\gamma_\ell\}_{\ell=1}^p$.

Define $N = pK$. For $i = 1 \dots N$,

Set $\ell = \text{mod}(i, p) + 1$

$x_i = x_{i-1} - \gamma_\ell U_\ell \nabla^{(\ell)} f(x_{i-1})$

Theorem 3.13. *Given a number of blocks p , a nonnegative constant L , and a function $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, let $\{x^{(n)}\}_{n=1,\dots,N}$ with $N = pK$ be the iterates generated by (CCD) applied to f , using step sizes $\gamma_\ell = \gamma$ for all $\ell \in \{1, \dots, p\}$, with $\gamma \leq \frac{1}{L}$.*

Then under assumptions of Setting ALL:

$$f(x_{pK}) - f(x_*) \leq \frac{4}{\gamma}(1 + p\gamma^2 L^2) \frac{p}{pK + 8} R_a^2. \quad (3.14)$$

Proof. Thanks to Theorem 3.9, we have the inclusion $\mathcal{F}_{0,L}(\mathbb{R}^d) \subset \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, where $\mathbf{L} = (L, \dots, L)$. We follow the proof of [14, Theorem 3.6], specifying it for the case $\mathbf{L} = (L, \dots, L)$ and replacing the inequalities (3.10) with those from Setting ALL:

$$\|x_{pk} - x_*\| \leq R_a, \quad \forall k \in \{1, \dots, K\}.$$

We thus obtain the rate of [14, Theorem 3.6] with $\bar{L}_{\min} = \bar{L}_{\max} = \frac{1}{\gamma}$ (since the step sizes in [14] are expressed in the form $\frac{1}{L_\ell}$ for each block, where $\bar{L}_\ell$ is viewed as an approximation of the smoothness constant L_ℓ of the ℓ^{th} block of coordinates). $\square$

Remark 3.14. The upper bound of Theorem 3.13 is specified for the functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$ but as explain in the proof the bound is also valid for the functional class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ with $\mathbf{L} = (L, \dots, L)$.

We also consider the cyclic alternating minimization algorithm (AM), where, at each step, an exact minimization is performed along the selected block of coordinates instead of applying a partial gradient step.

Algorithm 8 Alternating minimization (AM)

Input function f defined over $\mathbb{R}^d$ with p blocks, starting point $x_0 \in \mathbb{R}^d$, number of cycles K .
Define $N = pK$. For $i = 1 \dots N$,
Set $\ell = \text{mod}(n, p) + 1$
 $x_i = \arg \min_{z=x_{i-1}+U_\ell \Delta x^{(\ell)}, \Delta x^{(\ell)} \in \mathbb{R}^{d_\ell}} f(z)$

Theorem 3.15. Given a nonnegative constant L and a function $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, let $\{x_i\}_{i \in \{1, \dots, N\}}$, with $N = 2K$, be the iterates generated by K cycles of the 2-block (AM) algorithm applied to f . Then under the assumptions of Setting ALL:

$$f(x_{pK}) - f(x_*) \leq \frac{2LR_a^2}{K-1}.$$

Proof. A similar reasoning applies as in the previous theorem for (CCD). We follow the proof in [14], specifying it for the case $\mathbf{L} = (L, L)$ and replacing

the inequalities (3.10) with those from Setting ALL:

$$\|x_{pk} - x_*\| \leq R_a, \quad \forall k \in \{1, \dots, K\}.$$

□

Remark 3.16. Similar to the upper bound for (CCD), the upper bound of Theorem 3.15 is stated for the functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$ but remains valid for the broader class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ with $L = (L, \dots, L)$.

The choice of considering cyclic coordinate descent (CCD) and alternating minimization (AM) is motivated by the fact that these algorithms are among the most widely used and extensively studied within the class of block coordinate-wise descent methods. We now introduce a third less standard deterministic fixed-step first-order algorithm which is a cyclic version of the accelerated random coordinate descent algorithm from [36]. We denote this algorithm by (CACD). Our interest with (CACD) lies in comparing its worst-case performance to that of its random version (RACD), which features a proven accelerated convergence rate on average.

Algorithm 9 Cyclic Accelerated Coordinate Descent (CACD)

Input function f defined over $\mathbb{R}^d$ with p blocks, starting points $x_0 = z_0 \in \mathbb{R}^d$, $\theta_0 = \frac{1}{p}$, number of cycles K , step-size $\{\gamma_\ell\}_{\ell=1}^p$

Define $N = pK$. For $i = 1 \dots N$

 Set $\ell = \text{mod}(i, p) + 1$

$y_{i-1} = (1 - \theta_{i-1})x_{i-1} + \theta_{i-1}z_{i-1}$

$z_i = z_{i-1} - \frac{\gamma_\ell}{p\theta_{i-1}} U_\ell \nabla^{(\ell)} f(y_{i-1})$

$x_i = y_{i-1} + p\theta_{i-1}(z_i - z_{i-1})$

$\theta_i = \frac{\sqrt{\theta_{i-1}^4 + 4\theta_{i-1}^2 - \theta_{i-1}^2}}{2}$

Algorithm 10 Random Accelerated Coordinate Descent (RACD)

Input function f defined over $\mathbb{R}^d$ with p blocks, starting points $x_0 = z_0 \in \mathbb{R}^d$, $\theta_0 = \frac{1}{p}$, number of steps N , step-size $\{\gamma_\ell\}_{\ell=1}^p$

For $i = 1 \dots N$

Sample uniformly at random $\ell \in \{1, \dots, p\}$

$$y_{i-1} = (1 - \theta_{i-1})x_{i-1} + \theta_{i-1}z_{i-1}$$

$$z_i = z_{i-1} - \frac{\gamma_\ell}{p\theta_{i-1}} U_\ell \nabla^{(\ell)} f(y_{i-1})$$

$$x_i = y_{i-1} + p\theta_{i-1}(z_i - z_{i-1})$$

$$\theta_i = \frac{\sqrt{\theta_{i-1}^4 + 4\theta_{i-1}^2 - \theta_{i-1}^2}}{2}$$

Theorem 3.17. [36, Theorem 3] After N steps of the p -block (RACD) algorithm with step-sizes $\gamma_\ell = \frac{1}{L}$, $\forall \ell \in \{1, \dots, p\}$ on a function f that belongs to $\mathcal{F}_{0,L}(\mathbb{R}^d)$, we have that:

$$\mathbb{E}[f(x_N) - f^*] \leq \frac{4p^2}{(N-1+2p)^2} R^2$$

with $R = \left(1 - \frac{1}{p}\right) (f(x_0) - f^*) + \frac{1}{2} \|x_0 - x_*\|_L^2$, where $\|x_0 - x_*\|_L^2 = L \|x_0 - x_*\|^2$.

Proof. With the notations of Theorem 3 in [36], taking $\tau = 1$, it is straightforward to show that if f belongs to $\mathcal{F}_{0,L}(\mathbb{R}^d)$, f verifies the (ESO) condition [36, Assumption 1] with $v_\ell = L$, $\forall \ell \in \{1, \dots, p\}$, thus the proof of Theorem 3 in [36] remains valid in our case and we obtain the desired result. $\square$

Remark 3.18. Note that algorithms (CCD) and (CACD) are fixed-step first-order BCD algorithms as defined by (3.8), allowing a convex PEP formulation to be directly constructed following the approach in the previous section. In contrast, the alternating minimization algorithm (AM) does not fall within the framework of (3.8) since its iterates cannot be expressed as a linear combination of the initial iterate x_0 and past gradients g_i . However, a convex PEP can still be formulated for this algorithm. Specifically, defining a PEP for (AM) requires considering the matrices:

$$P^{(\ell)} = [g_0^{(\ell)}, \dots, g_N^{(\ell)}, x_0^{(\ell)}, \dots, x_N^{(\ell)}],$$

and the corresponding Gram matrix:

$$G^{(\ell)} = P^{(\ell)\top} P^{(\ell)} \in \mathbb{S}^{2N+2}.$$

Then, for all $\ell \in \{1, \dots, p\}$ and $i \in \{0, \dots, N\}$, we have:

$$x_i^{(\ell)} = P^{(\ell)} h_i, \quad \text{with } h_i = e_{N+2+i} \in \mathbb{R}^{2N+2},$$

and

$$g_i^{(\ell)} = P^{(\ell)} u_i, \quad \text{with } u_i = e_{i+1} \in \mathbb{R}^{2N+2},$$

where $\{e_i\}_{i \in \{1, \dots, 2N+2\}}$ is the canonical basis of $\mathbb{R}^{2N+2}$. The update step in (AM), given by:

$$x_i = \operatorname{argmin}_{z=x_{i-1}+U_\ell \Delta x^{(\ell)}, \Delta x^{(\ell)} \in \mathbb{R}^{d_\ell}} f(z),$$

can be modeled in a PEP through the conditions:

$$x_i^{(s)} = x_{i-1}^{(s)}, \quad \forall s \in \{1, \dots, p\}, s \neq \ell,$$

and

$$g_i^{(\ell)} = P^{(\ell)} u_i = 0.$$

The interpolation conditions, performance criterion, and initial condition are handled in the same manner as in the previous section, thereby demonstrating how to formulate a convex PEP for (AM).

Now that we have introduced the BCD algorithms to be analyzed, along with the corresponding settings, we present the numerical results on their worst-case convergence obtained using our PEP framework. For simplicity, we restrict our numerical experiments to the case of two blocks ($p = 2$) and assume a constant step size across all coordinate blocks, i.e., $\gamma_1 = \gamma_2 = \gamma$.

Let $\mathcal{W}_L^{\text{ALG}(\gamma)}(p, K, R)$ denote the worst-case performance of the fixed-step BCD algorithm $\text{ALG}(\gamma)$ with step sizes γ_ℓ over the functional class $\mathcal{F}_{0,L}(\mathbb{R}^d)$. Similarly, let $\mathcal{W}_{\text{coord},L}^{\text{ALG}(\gamma_\ell)}(p, K, R)$ represent the worst-case performance of $\text{ALG}(\gamma_\ell)$ with step sizes γ_ℓ over the functional class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. Using a proof similar to [96, Section 4.2.5] (same class of functions, step sizes and similar initial conditions as in [96, Section 4.2.5] considered here), the following homogeneity properties hold for both our settings ALL and INIT:

$$\mathcal{W}_L^{\text{CCD}(\frac{\gamma}{L})}(p, K, R) = LR^2 \mathcal{W}_1^{\text{CCD}(\gamma)}(p, K, 1)$$

$$\mathcal{W}_L^{\text{AM}}(p, K, R) = LR^2 \mathcal{W}_1^{\text{AM}}(p, K, 1)$$

$$\mathcal{W}_L^{ACD(\frac{\gamma}{L})}(p, K, R) = LR^2 \mathcal{W}_1^{ACD(\gamma)}(p, K, 1)$$

We can thus, without loss of generality, fix the smoothness constants and the bounds on the initial distance to optimality as $L_1 = L_2 = 1$ and $R_a = R_i = 1$.

The upper and lower bounds presented in the remainder of this section are thus those on $\mathcal{W}_{\text{coord},(1,1)}^{\text{ALG}(\gamma)}(2, K, 1)$ given by Theorem 3.9.

$$\mathcal{W}_1^{\text{ALG}(\gamma)}(2, K, 1) \leq \mathcal{W}_{\text{coord},(1,1)}^{\text{ALG}(\gamma)}(2, K, 1) \leq \mathcal{W}_2^{\text{ALG}(\gamma)}(2, K, 1) \quad (3.15)$$

Since a $\mathcal{O}(\frac{1}{K})$ rate of convergence is expected in most cases, we often show the evolution of our performance criterion multiplied by K to facilitate the analysis.

Algorithm (CCD). In Figure 3.1, we provide the upper and lower bounds in (3.15), obtained thanks to Theorem 3.9, for the worst case of (CCD) with a constant step size $\gamma = \frac{1}{2}$. We choose this step size to ensure the convergence of our upper bound, for which we consider the class of 2-smooth convex functions. Our bounds are sublinear and improve the analytical bound given in (3.13) by approximately a multiplicative factor of 10. The lower bound in Figure 3.1 suggests that our upper bound on the worst case of (CCD) over the class of coordinate-wise smooth convex functions cannot be significantly improved. The experiments presented in Figure 3.1 are performed under the assumptions of Setting ALL. In order to illustrate the flexibility and usefulness of our framework, we provide in addition bounds for the 2-block (CCD) in Setting INIT. In Figure 3.2, we compare the upper bounds obtained for both settings for block coordinate descent. Our results show that the convergence of cyclic block coordinate descent can also be established under the weaker assumption of Setting INIT, and the results suggest that the stronger assumptions made in Setting ALL do not yield a significant improvement in terms of performance as we only start to observe a slightly better performance under Setting ALL after around 25 steps. We also provide bounds for (CCD) with the residual gradient norm as a performance criterion. In Figure 3.3, we show the reciprocal of the upper and lower bounds we obtain using PEP on the residual gradient, which suggests that the squared residual gradient norm converges at a rate of $\mathcal{O}(1/K^2)$, which is faster than the convergence of the function accuracy (a similar situation holds for the full step gradient descent).

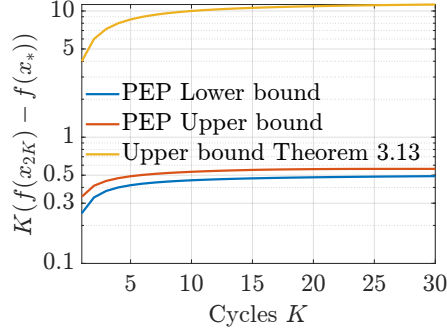


Fig. 3.1 Comparison of the upper and lower bounds from Theorem 3.9, computed numerically using our PEP framework, with the analytical bound from Theorem 3.13, adapted from [14] and scaled by the number of cycles K , for the 2-block (CCD) with step size $\gamma = 1/2$ in Setting ALL.

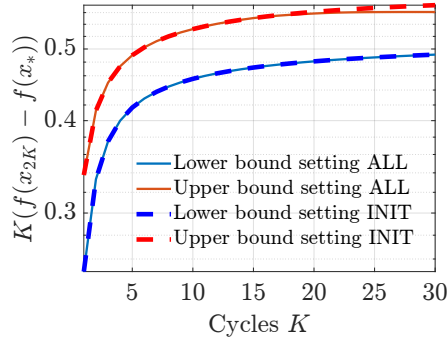


Fig. 3.2 Comparison of the upper and lower bounds from Theorem 3.9, numerically computed using our PEP framework and scaled by the number of cycles K , for the 2-block (CCD) with step size $\gamma = 1/2$ in Settings ALL (solid lines) and INIT. (dashed lines).

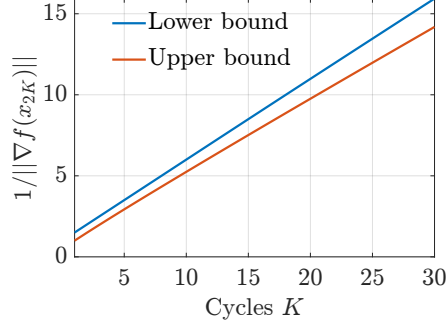


Fig. 3.3 Reciprocal of the upper and lower bounds from Theorem 3.9, numerically computed using our PEP framework for the residual gradient norm for the 2-block (CCD) under Setting ALL.

Algorithm (AM). In Figure 3.4, we provide upper and lower bounds for the alternating minimization algorithm in Setting ALL and compare them to the bound provided in Theorem 3.15 (adapted from [14, Theorem 5.2]). Figure 3.4 shows that our PEP-based approach improves the bound in Theorem 3.15 by a multiplicative factor of two.

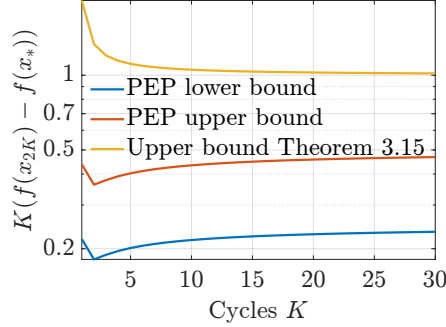


Fig. 3.4 Comparison of the upper and lower bounds from Theorem 3.9, computed using our PEP framework, with the analytical bound from Theorem 3.15 (adapted from [14, Theorem 5.2]) and scaled by the number of cycles K , for the 2-block (AM) in Setting ALL.

Algorithm (CACD). We analyze (CACD) over the class of 1-smooth convex functions. By Theorem 3.9, this also provides a lower bound on the worst-case performance for the function class $\mathcal{F}_{0,(1,\dots,1)}^{\text{coord}}(\mathbb{R}^d)$. We choose a step size $\gamma_\ell = 1, \forall \ell \in \{1, \dots, p\}$, allowing us to compare our PEP-generated bound for (CACD) with the convergence bound of (RACD) given in Theo-

rem 3.17. In Figure 3.5, we plot the reciprocal of our lower bound as well as a numerical linear fit, suggesting that the convergence rate of (CACD) is of order $\mathcal{O}(1/K)$. No acceleration is observed, as the rate remains slower than $\mathcal{O}(1/K^2)$, which holds for the random (RACD). A numerical fit of our lower bound gives

$$\mathcal{W}_1^{\text{CACD}(1)}(2, K, 1) \approx \frac{0.0945}{K - 3.1026}.$$

This suggests that randomness in the choice of the blocks of coordinates to update plays a key role in accelerating BCD algorithms. To investigate this further, we use the convex PEP framework for random BCD algorithms, as formulated in Problem (PEP-rdn-Smooth), to compute the exact worst-case expected performance, denoted by $\mathcal{W}^{(\text{RACD})}$, for (RACD) with step sizes $\gamma_\ell = 1, \forall \ell \in \{1, \dots, p\}$, over the class of 1-smooth convex functions. Since analyzing N steps of the 2-block (RACD) requires considering 2^N possible sequences of updates, we restrict our study to $N = 4$. Table 3.1 reports the worst-case performance for each deterministic sequence of block coordinate updates of length $N = 4$ and the worst-case after $N = 4$ steps of the random algorithm (RACD), given by $\mathcal{W}^{(\text{RACD})} = 0.102$. Notably, all deterministic sequences in Table 3.1 yield a worse worst-case bound than $\mathcal{W}^{(\text{RACD})}$, except for cyclic updates (1 2 1 2 or 2 1 2 1) and mirror updates (1 2 2 1 or 2 1 1 2). A possible explanation is that the worst-case expected performance $\mathcal{W}^{(\text{RACD})}$ includes clearly suboptimal update sequences, such as always updating only a single block of coordinates. To confirm this, we computed the worst-case expected performance when restricting the possible sequence of blocks to only the two best deterministic strategies (cyclic updates and mirror updates). This yield $\mathcal{W}_{\text{best}} = 0.094$, slightly better than all the best deterministic cases. Although the preliminary results in Table 3.1 are not entirely conclusive, they suggest, together with Figure 3.5, that (CACD) is indeed slower than its random counterpart.

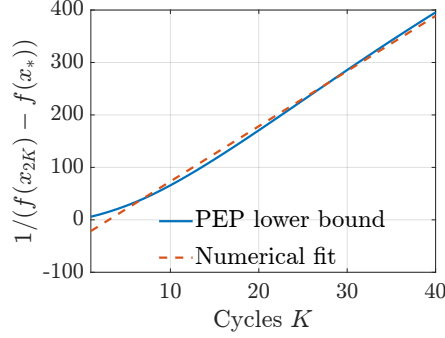


Fig. 3.5 Reciprocal of the lower bound given by Theorem 3.9 and numerically computed with our PEP framework and its linear numerical fit for the 2-block (CACD) in Setting INIT, which indicates that the convergence rate is slower than $\mathcal{O}(1/K^2)$.

Block choice type	Ordered block choices	Worst-case
Cyclic ($= \mathcal{W}^{\text{CACD}}$)	1 2 1 2 or 2 1 2 1	0.098
Fixed choice	1 2 2 1 or 2 1 1 2	0.100
Fixed choice	1 2 1 1 or 2 1 2 2	0.135
Fixed choice	1 1 2 1 or 2 2 1 2	0.152
Fixed choice	1 2 2 2 or 2 1 1 1	0.155
Fixed choice	1 1 2 2 or 2 2 1 1	0.1667
Fixed choice	1 1 1 2 or 2 2 2 1	0.188
Fixed choice	1 1 1 1 or 2 2 2 2	0.500
Random ($= \mathcal{W}^{\text{RACD}}$)	at each step: pick 1 or 2	0.102
Random best	at each cycle: pick 1 2 or 2 1	0.094

Table 3.1 Comparison of the worst-case performance of (RACD) and the worst-case performance of all deterministic variants for $(p = 2, K = 2)$.

3.5 A convex Performance Estimation Problem framework for the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$

To compute tighter upper bounds on the convergence of BCD algorithms over the class of functions $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ without relying on the double inclusion that relates it to the classes $\mathcal{F}_{0,L_{\min}}(\mathbb{R}^d)$ and $\mathcal{F}_{0,L_{\text{sum}}}(\mathbb{R}^d)$ (see Theorem 3.9), we now derive a convex PEP formulation for fixed-step first-order BCD algorithms directly over the class of block coordinate-wise smooth

functions $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. This derivation follows closely the construction of the convex PEP for fixed-step first-order BCD algorithms over the class of smooth convex functions $\mathcal{F}_{0,L}(\mathbb{R}^d)$ presented in Section 3.3. The key difference in considering the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ instead of $\mathcal{F}_{0,L}(\mathbb{R}^d)$ lies in replacing the $\mathcal{F}_{0,L}(\mathbb{R}^d)$ -interpolability constraint on the set of decision variables $\mathcal{S}_N$ with a $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolability constraint. This requires identifying appropriate interpolation conditions to enforce the new interpolability constraint. In the following theorem, we provide necessary interpolation conditions for the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$:

Theorem 3.19. *Given I a finite index set and a number of blocks of coordinates p . If the set $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I} \subset \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$ is $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolable, then*

$$\forall i, j \in I, \forall \ell \in \{1, \dots, p\}$$

$$f_i \geq f_j + \sum_{\ell=1}^p \langle g_j^{(\ell)}, x_i^{(\ell)} - x_j^{(\ell)} \rangle + \frac{1}{2L_\ell} \|g_i^{(\ell)} - g_j^{(\ell)}\|^2. \quad (3.16)$$

Proof. If $\mathcal{S}$ is $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolable then any function $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ that interpolates $\mathcal{S}$ satisfies, by Theorem 3.7, the conditions (3.5) over the entire space $\mathbb{R}^d$ and in particular over the set $\mathcal{S}$ which yields the interpolation conditions (3.16). $\square$

Before presenting the formulation of the PEP for the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, let us first discuss the sufficiency of the interpolation conditions (3.16). In the next theorem, we prove that for sets I with cardinality at most 2, these interpolation conditions are sufficient:

Theorem 3.20. *A set $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i=1,\dots,N} \subset \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$, with $N \leq 2$, is $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ -interpolable if and only if it verifies the conditions (3.16).*

Proof. The necessity follows from Theorem 3.19. We establish sufficiency by constructing an explicit example of an interpolating function defined via its Fenchel conjugate. Note that the following propositions are equivalent:

1. Function f interpolates the set $\{(x_i, g_i, f_i)\}_{i=1,2}$.
2. The Fenchel conjugate f^* of function f interpolates the set $\{(g_i, x_i, \langle x_i, g_i \rangle - f_i)\}_{i=1,2}$.

This follows from Young's equality (see, for instance, [96, Theorem 2.21]). We assume, without loss of generality, that

$$f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle \geq f_2 - f_1 - \langle g_1, x_2 - x_1 \rangle.$$

Since the problem is symmetric in (f_1, g_1, x_1) and (f_2, g_2, x_2) , the remainder of the proof remains valid by switching the roles of the variables if the previous inequality is reversed. Consider the function:

$$f^*(g) = \begin{cases} \phi(\lambda) & \text{if } g = \lambda g_2 + (1 - \lambda)g_1, \lambda \in [0, 1], \\ \infty & \text{otherwise.} \end{cases} \quad (3.17)$$

where:

$$\phi(\lambda) = (f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle)\lambda^2 + \langle x_1, g_2 - g_1 \rangle\lambda + \langle g_1, x_1 \rangle - f_1.$$

This function is only finite on the segment between g_1 and g_2 . It is easy to verify that:

$$f^*(g_1) = \phi(0) = \langle g_1, x_1 \rangle - f_1, \quad f^*(g_2) = \phi(1) = \langle g_2, x_2 \rangle - f_2.$$

Let us now prove that x_1 and x_2 are subgradients of f^* at the points g_1 and g_2 , respectively which means that for any $g \in \mathbb{R}^d$

$$f^*(g) \geq f^*(g_1) + \langle x_1, g - g_1 \rangle,$$

$$f^*(g) \geq f^*(g_2) + \langle x_2, g - g_2 \rangle.$$

Consider the first case where $g \in \mathbb{R}^d$ and $g \neq \lambda g_2 + (1 - \lambda)g_1$ for any $\lambda \in [0, 1]$. In this case, we have $f^*(g) = +\infty$, and the subgradient inequalities are trivially satisfied. Now, consider the case where $g = \lambda g_2 + (1 - \lambda)g_1$. Since $(f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle) \geq 0$, ϕ is a convex quadratic function and we have:

$$\phi(\lambda) \geq \phi(0) + \phi'(0)\lambda,$$

which is equivalent to:

$$f^*(g) \geq f^*(g_1) + \langle x_1, g_2 - g_1 \rangle\lambda.$$

Simple computations based on $g = \lambda g_2 + (1 - \lambda)g_1$ give:

$$\langle x_1, g_2 - g_1 \rangle \lambda = \langle x_1, g - g_1 \rangle,$$

which then leads to:

$$f^*(g) \geq f^*(g_1) + \langle x_1, g - g_1 \rangle.$$

Thus, x_1 is a subgradient of f^* at g_1 . Similarly, from the convexity of ϕ , we obtain:

$$\phi(\lambda) \geq \phi(1) + \phi'(1)(\lambda - 1),$$

which is equivalent to:

$$f^*(g) \geq f^*(g_2) + (2(f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle) + \langle x_1, g_2 - g_1 \rangle)(\lambda - 1).$$

Since we assume that:

$$f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle \geq f_2 - f_1 - \langle g_1, x_2 - x_1 \rangle,$$

it follows by adding $f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle$ on both sides that:

$$2(f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle) \geq \langle x_2 - x_1, g_2 - g_1 \rangle. \quad (3.18)$$

This implies:

$$f^*(g) \geq f^*(g_2) + (\langle x_2 - x_1, g_2 - g_1 \rangle + \langle x_1, g_2 - g_1 \rangle)(\lambda - 1),$$

which is equivalent to

$$f^*(g) \geq f^*(g_2) + \langle x_2, g_2 - g_1 \rangle(\lambda - 1).$$

Since:

$$\langle x_2, g_2 - g_1 \rangle(\lambda - 1) = \langle x_2, g - g_2 \rangle,$$

we obtain:

$$f^*(g) \geq f^*(g_2) + \langle x_2, g - g_2 \rangle,$$

and x_2 is a subgradient at g_2 . Hence function f^* interpolates the set $\{(g_i, x_i, \langle x_i, g_i \rangle - f_i)\}_{i=1,2}$, which implies that $f = (f^*)^*$, defined as:

$$f(x) = \sup_{g \in \mathbb{R}^d} \langle x, g \rangle - f^*(g), \quad \forall x \in \mathbb{R}^d,$$

also interpolates the set $\{(x_i, g_i, f_i)\}_{i=1,2}$. By definition of f^* , f takes finite values for all $x \in \mathbb{R}^d$, and for any $x \in \mathbb{R}^d$, there exists $g_x = \lambda_x g_2 + (1 - \lambda_x)g_1 \in \mathbb{R}^d$ such that:

$$f(x) = \langle x, g_x \rangle - f^*(g_x). \quad (3.19)$$

By Young's equality (see [96, Theorem 2.21]), we have $g_x \in \partial f(x)$ and $x \in \partial f^*(g_x)$. To complete the proof, it remains to show that $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. To do so, we first prove that the functions $g \mapsto f^*(g) - \frac{1}{2L_\ell} \|g^{(\ell)}\|^2$ are convex and show that this implies $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. Simple computations yield:

$$h_\ell(g) = f^*(g) - \frac{1}{2L_\ell} \|g^{(\ell)}\|^2 = \begin{cases} \psi_\ell(\lambda), & \text{if } g = \lambda g_2 + (1 - \lambda)g_1, \lambda \in [0, 1] \\ +\infty, & \text{otherwise,} \end{cases}$$

where:

$$\begin{aligned} \psi_\ell(\lambda) = & \left(f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle - \frac{1}{2L_\ell} \|g_2^{(\ell)} - g_1^{(\ell)}\|^2 \right) \lambda^2 \\ & + \left(\langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g_2 - g_1 \rangle \right) \lambda \\ & + \langle g_1, x_1 \rangle - f_1 - \frac{1}{2L_\ell} \|g_1^{(\ell)}\|^2. \end{aligned}$$

Since the set $\{(x_i, g_i, f_i)\}_{i=1,2}$ satisfies the interpolation conditions (3.16), the term

$$f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle - \frac{1}{2L_\ell} \|g_2^{(\ell)} - g_1^{(\ell)}\|^2$$

is nonnegative. Thus, ψ_ℓ is a convex quadratic function, implying that h_ℓ is convex. For all $x \in \mathbb{R}^d$ and g_x as defined in (3.19), we have that if $g_x \notin \{g_1, g_2\}$, f^* is differentiable and $x - \frac{1}{L_\ell} U_\ell g_x^{(\ell)}$ is the gradient of h_ℓ at g_x . Moreover, $x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}$ and $x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}$ are respectively subgradients of h_ℓ at g_1 and g_2 . Indeed, consider the first case where $g \in \mathbb{R}^d$ and $g \neq \lambda g_2 + (1 - \lambda)g_1$ for any $\lambda \in [0, 1]$. In this case, we have $h_\ell(g) = +\infty$, and the subgradient inequalities

$$h_\ell(g) \geq h_\ell(g_1) + \langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g - g_1 \rangle,$$

$$h_\ell^*(g) \geq h_\ell(g_2) + \langle x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}, g - g_2 \rangle$$

are trivially satisfied. Now, consider the case where $g = \lambda g_2 + (1 - \lambda)g_1$. Since ϕ is a convex quadratic function, we have:

$$\psi_\ell(\lambda) \geq \psi_\ell(0) + \psi'_\ell(0)\lambda,$$

which is equivalent to:

$$h_\ell(g) \geq h_\ell(g_1) + \langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g_2 - g_1 \rangle \lambda.$$

Simple computations give:

$$\langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g_2 - g_1 \rangle \lambda = \langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g - g_1 \rangle,$$

which then leads to:

$$h_\ell(g) \geq h_\ell(g_1) + \langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g - g_1 \rangle.$$

Thus, $x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}$ is a subgradient of h_ℓ at g_1 . Similarly, from the convexity of ψ_ℓ , we obtain:

$$\psi_\ell(\lambda) \geq \psi_\ell(1) + \psi'_\ell(1)(\lambda - 1),$$

which is equivalent to:

$$\begin{aligned} h_\ell(g) \geq h_\ell(g_2) + (2(f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle - \frac{1}{2L_\ell} \|g_2^{(\ell)} - g_1^{(\ell)}\|^2) + \\ \langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g_2 - g_1 \rangle)(\lambda - 1). \end{aligned}$$

Since we assume that:

$$f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle \geq f_2 - f_1 - \langle g_1, x_2 - x_1 \rangle,$$

it follows that:

$$2(f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle) \geq \langle x_2 - x_1, g_2 - g_1 \rangle. \quad (3.20)$$

This implies:

$$\begin{aligned}
2(f_1 - f_2 - \langle g_2, x_1 - x_2 \rangle - \frac{1}{2L_\ell} \|g_2^{(\ell)} - g_1^{(\ell)}\|^2) + \langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g_2 - g_1 \rangle \geq \\
\langle x_2 - x_1, g_2 - g_1 \rangle - \frac{1}{2L_\ell} \|g_2^{(\ell)} - g_1^{(\ell)}\|^2 + \langle x_1 - \frac{1}{L_\ell} U_\ell g_1^{(\ell)}, g_2 - g_1 \rangle = \\
\langle x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}, g_2 - g_1 \rangle
\end{aligned}$$

which gives us

$$h_\ell(g) \geq h_\ell(g_2) + \langle x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}, g_2 - g_1 \rangle (\lambda - 1).$$

Since:

$$\langle x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}, g_2 - g_1 \rangle (\lambda - 1) = \langle x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}, g - g_2 \rangle,$$

we obtain:

$$h_\ell(g) \geq h_\ell(g_2) + \langle x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}, g - g_2 \rangle,$$

which proves that $x_2 - \frac{1}{L_\ell} U_\ell g_2^{(\ell)}$ is a subgradient of h_ℓ at g_2 . By the convexity of h_ℓ , for all $\ell \in \{1, \dots, p\}$ and for all $x, y \in \mathbb{R}^d$, we obtain:

$$f^*(g_y) - \frac{1}{2L_\ell} \|g_y^{(\ell)}\|^2 \geq f^*(g_x) - \frac{1}{2L_\ell} \|g_x^{(\ell)}\|^2 + \left\langle x - \frac{1}{L_\ell} U_\ell g_x^{(\ell)}, g_y - g_x \right\rangle.$$

By substituting the equalities:

$$f(x) + f^*(g_x) = \langle x, g_x \rangle,$$

$$f(y) + f^*(g_y) = \langle y, g_y \rangle,$$

the previous inequality becomes, for all $\ell \in \{1, \dots, p\}$ and for all $x, y \in \mathbb{R}^d$:

$$f(x) - f(y) \geq \langle g_y, x - y \rangle + \frac{1}{2L_\ell} \left(\|g_y^{(\ell)}\|^2 - \|g_x^{(\ell)}\|^2 - 2\langle U_\ell g_x^{(\ell)}, g_y - g_x \rangle \right).$$

From the definition of the selection matrix U_ℓ , we have:

$$\langle U_\ell g_x^{(\ell)}, g_y \rangle = \langle g_x^{(\ell)}, g_y^{(\ell)} \rangle,$$

and:

$$\langle U_\ell g_x^{(\ell)}, g_x \rangle = \|g_x^{(\ell)}\|^2.$$

Thus, for all $\ell \in \{1, \dots, p\}$ and for all $x, y \in \mathbb{R}^d$, we obtain:

$$f(x) - f(y) \geq \langle g_y, x - y \rangle + \frac{1}{2L_\ell} \|g_x^{(\ell)} - g_y^{(\ell)}\|^2.$$

Since $g_x \in \partial f(x)$ and $g_y \in \partial f(y)$, Theorem 3.7 ensures that f belongs to $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, which concludes the proof. $\square$

We show in the next theorem that for sets of cardinality $N > 2$, the interpolation conditions (3.16) are not sufficient. We establish this by providing a counterexample for $N = 3$.

Theorem 3.21 (Insufficiency of interpolation conditions for $N > 2$). *For any set of the form $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i=1,\dots,N}$ with $N > 2$, the interpolation conditions (3.16) are not sufficient.*

Proof. We construct a counterexample with $N = 3$ that satisfies (3.16) but is not interpolable by a function $f \in \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. Consider the set:

Points x_i	Gradients g_i	Function Values f_i
$x_1 = (-1, 0)^T$	$g_1 = (-1, 0)^T$	$f_1 = \frac{1}{2}$
$x_2 = (0, 0)^T$	$g_2 = (0, -1)^T$	$f_2 = 0$
$x_3 = (1, 0)^T$	$g_3 = (1, 0)^T$	$f_3 = \frac{1}{2}$

Table 3.2 A set $\mathcal{S}$ satisfying (3.16) but not interpolable.

The set $\mathcal{S}$ satisfies (3.16) for $\mathbf{L} = (1, 1)$. Suppose that there exists a function $f \in \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^2)$ that interpolates $\mathcal{S}$. For any point $x = (-1, y)$, the convexity of f implies that

$$f(x) \geq f(x_1) + \langle g_1, x - x_1 \rangle,$$

which is equivalent to

$$f(-1, y) \geq \frac{1}{2}.$$

Similarly for the points of the form $x = (1, y)$, we have that

$$f(x) \geq f(x_3) + \langle g_3, x - x_3 \rangle,$$

which is equivalent to:

$$f(1, y) \geq \frac{1}{2}.$$

By 1-smoothness of f along the second coordinate, for any point $x = (0, y)$ we have

$$f(0, y) \leq f(0, 0) + g_2^{(2)} y + \frac{y^2}{2} = -y + \frac{y^2}{2}.$$

Setting $y = 1$, we obtain from all the above:

$$f(-1, 1) \geq \frac{1}{2}, \quad f(0, 1) \leq -\frac{1}{2}, \quad f(1, 1) \geq \frac{1}{2}.$$

Define $h(x) = f(x, 1)$. Then h is a 1-smooth convex function satisfying:

$$h(-1) \geq \frac{1}{2}, \quad h(0) \leq -\frac{1}{2}, \quad h(1) \geq \frac{1}{2}.$$

Suppose $h'(0) \leq 0$, then since h is a 1-smooth convex function, we have that

$$h(1) \leq h(0) + h'(0)(1 - 0) + \frac{(1 - 0)^2}{2} \leq h(0) + \frac{1}{2} \leq 0,$$

which contradicts $h(1) \geq \frac{1}{2}$. Similarly, if $h'(0) \geq 0$, we have that:

$$h(-1) \leq h(0) + h'(0)(-1 - 0) + \frac{(-1 - 0)^2}{2} \leq h(0) + \frac{1}{2} \leq 0,$$

which contradicts $h(-1) \geq \frac{1}{2}$. Thus, an interpolating function $f \in \mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ for the set $\mathcal{S}$ cannot exist. $\square$

Remark 3.22. Theorems 3.20 and 3.21 may suggest that necessary and sufficient interpolation cannot be enforced by pairwise conditions but instead should result from conditions involving at least 3 points. Moreover, since the interpolation conditions (3.16) are generally only necessary, using them in a PEP framework introduces a relaxation, providing only a valid upper bound on the worst-case convergence of the BCD algorithm under analysis.

To conclude this section, we show how the interpolation conditions (3.16) can be used to derive a convex relaxation for the PEP of fixed-step first-order algorithms over the functional class $\mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. Using the definitions introduced in Section 3.3, the interpolation conditions (3.16) can be expressed for all $\ell \in \{1, \dots, p\}$ and $i, j \in I$ in the form:

$$F^\top(u_j - u_i) + \sum_{\ell=1}^p \text{Tr} \left(A_{i,j}^{(\ell)}(\alpha) G^{(\ell)} \right) + \text{Tr} \left(B_{i,j}^{(\ell)} G^{(\ell)} \right) \leq 0.$$

with:

$$2A_{i,j}^{(\ell)}(\alpha) = u_j(h_i^{(\ell)} - h_j^{(\ell)})^\top + (h_i^{(\ell)} - h_j^{(\ell)})u_j^\top,$$

$$2B_{i,j}^{(\ell)} = \frac{1}{L_\ell}(u_i - u_j)(u_i - u_j)^\top.$$

Replacing the interpolation conditions for the class $\mathcal{F}_{0,L}(\mathbb{R}^d)$ by the interpolation conditions for the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, we obtain:

$$w^{\text{coord}}(R, \mathbf{L}, \alpha_{i,k}, N, b, C^{(\ell)}) = \sup_{G^{(\ell)}, F} b^\top F + \sum_{\ell=1}^p \text{Tr}(C^{(\ell)} G^{(\ell)})$$

subject to:

$$F^\top(u_j - u_i) + \sum_{\ell=1}^p \text{Tr}(A_{i,j}^{(\ell)}(\alpha) G^{(\ell)}) +$$

$$\text{Tr}(B_{i,j}^{(\ell)} G^{(\ell)}) \leq 0, \quad \forall \ell \in \{1, \dots, p\} \quad \forall i, j \in I,$$

$$\sum_{\ell=1}^p \text{Tr}(A_R^{(\ell)} G^{(\ell)}) \leq R^2,$$

$$G^{(\ell)} \succeq 0, \quad \forall \ell \in \{1, \dots, p\}.$$

(SDP-coord-PEP)

This is a convex SDP whose optimal value provides a valid upper bound on the worst-case performance after N steps of the fixed-step first-order BCD algorithm applied to the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. Since the problem is dimension-independent (as the block dimensions d_ℓ do not appear in the formulation), the bound is valid for every $d \in \mathbb{N}$ and thus holds over the set of all $\mathbf{L}$ -block coordinate-wise smooth convex functions $\mathcal{F}_{0,L}^{\text{coord}} = \bigcup_{d \in \mathbb{N}} \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$:

Theorem 3.23. *Given a number of blocks of coordinates p , a number of steps N and a vector of nonnegative constants $\mathbf{L}$, consider the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ of $\mathbf{L}$ -block coordinate-wise smooth convex functions, the deterministic fixed-step first-order BCD method $\mathcal{M}_\alpha^{\text{coord}}$ as defined in (3.8), and a performance criterion $\mathcal{P}(F, G^{(\ell)}) = b^\top F + \sum_{\ell=1}^p \text{Tr}(C^{(\ell)} G^{(\ell)})$. For all $d \in \mathbb{R}^d$, a valid upper bound on the worst-case performance after N iterations of method $\mathcal{M}_\alpha^{\text{coord}}$ over the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ is given by the optimal value of Problem (SDP-coord-PEP).*

A dual formulation of Problem (SDP-coord-PEP), which could, for instance, be useful for deriving proofs of convergence theorems or for formulating the problem of designing optimized BCD algorithms (see Chapter 2), is

given by:

$$\inf_{\lambda_{\ell,i,j}, \tau} \tau R^2$$

subject to:

$$\tau A_R^{(\ell)} - C + \sum_{s=1}^p \sum_{i,j \in I} \lambda_{s,i,j} A_{i,j}^{(s)}(\alpha) + \sum_{i,j \in I} \lambda_{\ell,i,j} B_{i,j}^{(\ell)} \succeq 0,$$

$$\forall \ell \in \{1, \dots, p\},$$

$$b - \sum_{s=1}^p \sum_{i,j \in I} \lambda_{s,i,j} (u_j - u_i) = 0,$$

$$\lambda_{\ell,i,j} \geq 0, \quad \forall \ell \in \{1, \dots, p\}, \quad \forall i, j \in I,$$

$$\tau \geq 0.$$

(dual-SDP-PEP-coord)

Using the same reasoning as in the analysis of BCD algorithms over the functional classes $\mathcal{F}_{0,L}(\mathbb{R}^d)$ (Section 3.3), the PEP framework for deterministic BCD algorithms over the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ can be extended to handle random algorithms:

Theorem 3.24. *Given a number of block of coordinates p , a number of steps N and a vector of nonnegative constants $\mathbf{L}$. Consider the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ of L -block coordinate-wise smooth convex functions, the random fixed-step first-order BCD method $\mathcal{M}_{\alpha}^{\text{coord}}$ of the form (3.8), and a performance criterion $\mathcal{P}(F, G^{(\ell)}) = b^\top F + \sum_{\ell=1}^p \text{Tr}(C^{(\ell)} G^{(\ell)})$. For all $d \in \mathbb{R}^d$, a valid upper bound on the worst-case expected performance after N iterations of method $\mathcal{M}_{\alpha}^{\text{coord}}$ over the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$*

is given by the optimal value of the following semidefinite program:

$$\begin{aligned}
w^{\text{rdn-coord}}(R, L, \alpha, N, b_r, C_r^{(\ell)}) &= \sup_{G^{(\ell)}, F} \sum_{r=1}^{p^N} \mathbb{P}_r \left(b_r^\top F + \sum_{\ell=1}^p \text{Tr}(C_r^{(\ell)} G^{(\ell)}) \right) \\
&\text{subject to:} \\
&F^\top (u_j - u_i) \\
&+ \sum_{\ell=1}^p \text{Tr} \left(A_{i,j}^{(\ell)}(\alpha) G^{(\ell)} \right) + \text{Tr} \left(B_{i,j}^{(\ell)} G^{(\ell)} \right) \leq 0, \\
&\forall \ell \in \{1, \dots, p\}, \forall i, j \in \{0, 1, \dots, Np^N, *\}, \\
&\sum_{\ell=1}^p \text{Tr}(A_R^{(\ell)} G^{(\ell)}) \leq R^2, \\
&G^{(\ell)} \succeq 0, \quad \forall \ell \in \{1, \dots, p\}.
\end{aligned}$$

(PEP-rdn-coord)

where the variable $r \in \{1, \dots, p^N\}$ is the index of a possible sequence of block coordinate updates, each corresponding to a deterministic algorithm $\mathcal{M}_r^{\text{coord}}(\alpha_{i,k})$ (as explained in Section 3.3 for the functional classes $\mathcal{F}_{0,L}(\mathbb{R}^d)$ of smooth convex functions).

3.6 Worst-case Behavior of Cyclic Coordinate Descent

In this section, we establish several general results on the worst-case behavior of cyclic BCD algorithms. Specifically, we prove an invariance property of the worst-case performance with respect to the norm $\|\cdot\|_L$ introduced earlier, a lower bound on the worst-case performance of (CCD) over the class of coordinate-wise smooth convex functions—equal to the number of blocks times the worst-case performance of gradient descent over smooth convex functions—, an improved descent lemma for (CCD), and an upper bound on the residual gradient norm for the 2-block (CCD).

Usually, the convergence upper bounds derived for first-order algorithms in the literature are independent of the function dimension. Hence, recall that $\mathcal{F}_{0,L}^{\text{coord}} = \bigcup_{d \in \mathbb{N}} \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ is the set of all L -block coordinate-wise smooth convex functions, and $\mathcal{F}_{0,L} = \bigcup_{d \in \mathbb{N}} \mathcal{F}_{0,L}(\mathbb{R}^d)$ is the set of all L -smooth convex functions. We denote by $\mathcal{W}_L^{\text{CCD}(\gamma)}(p, K)$ the worst-case performance after K cycles of the p -block (CCD) algorithm with step sizes γ_ℓ over the set

of $\mathbf{L}$ -block coordinate-wise smooth functions $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}$:

$$\mathcal{W}_{\mathbf{L}}^{\text{CCD}(\gamma)}(p, K) = \max_{f \in \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}} \frac{f(x_{pK}) - f_*}{\|x_0 - x_*\|_{\mathbf{L}}^2}.$$

Similarly, we denote by $\mathcal{W}_{\mathbf{L}}^{\text{MGD}(\alpha)}(N)$ the worst-case performance after N steps of the memoryless gradient descent (MGD) with step sizes α_i over the set of L -smooth convex functions $\mathcal{F}_{0,L}$:

$$\mathcal{W}_{\mathbf{L}}^{\text{MGD}(\alpha)}(N) = \max_{f \in \mathcal{F}_{0,L}} \frac{f(x_N) - f_*}{\|x_0 - x_*\|_L^2},$$

where $\|x_0 - x_*\|_L^2 = L\|x_0 - x_*\|^2$.

We begin by proving a scale-invariance property for the worst-case behavior of cyclic coordinate descent (CCD) with respect to the scaled norm $\|\cdot\|_L$. Building on this result, we show that the p -block (CCD) algorithm has a convergence rate on coordinate smooth functions at least p times worse than that of the standard full gradient method on smooth convex functions.

Theorem 3.25. *Given a vector of nonnegative constants $\mathbf{L}$, the worst-case performance after K cycles of p -block cyclic coordinate descent (CCD) with step sizes of the form $\{\frac{\gamma_\ell}{L_\ell}\}_{\ell=1}^p$ over the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}$ satisfies the following homogeneity property with respect to the vector of block coordinate-wise smoothness constants $\mathbf{L}$:*

$$\mathcal{W}_{\mathbf{L}}^{\text{CCD}(\frac{\gamma_\ell}{L_\ell})}(p, K) = \mathcal{W}_{(1,\dots,1)}^{\text{CCD}(\gamma_\ell)}(p, K)$$

Proof. Given a function f that belongs to $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, define the function $\tilde{f} : x \mapsto f\left(\frac{1}{\sqrt{L_1}}x^{(1)}, \dots, \frac{1}{\sqrt{L_p}}x^{(p)}\right)$. Then, $\tilde{f}$ belongs to $\mathcal{F}_{0,(1,\dots,1)}^{\text{coord}}(\mathbb{R}^d)$. Indeed, since f is convex, $\tilde{f}$ is also convex, and we have:

$$\nabla^{(\ell)}\tilde{f}(x) = \frac{1}{\sqrt{L_\ell}}\nabla^{(\ell)}f\left(\frac{1}{\sqrt{L_1}}x^{(1)}, \dots, \frac{1}{\sqrt{L_p}}x^{(p)}\right),$$

which, by the $\mathbf{L}$ -block coordinate-wise smoothness of f , implies that for all

$x \in \mathbb{R}^d$ and $h^{(\ell)} \in \mathbb{R}^{d_\ell}$,

$$\begin{aligned} \|\nabla^{(\ell)} \tilde{f}(x + U_\ell h^{(\ell)}) - \nabla^{(\ell)} \tilde{f}(x)\| &\leq \frac{1}{\sqrt{L_\ell}} L_\ell \left\| \frac{1}{\sqrt{L_\ell}} h^{(\ell)} \right\| \\ &= \|h^{(\ell)}\|. \end{aligned}$$

Thus, $\tilde{f}$ is $(1, \dots, 1)$ -block coordinate-wise smooth. Now, denote by x_i the iterates of $\text{CCD}(\frac{\gamma_\ell}{L_\ell})$ on f :

$$x_i = x_{i-1} - \frac{\gamma_\ell}{L_\ell} U_\ell \nabla^{(\ell)} f(x_{i-1}),$$

where ℓ is chosen cyclically as $\ell = \text{mod}(i, p) + 1$. Let x_* be a minimizer of f , and define $\tilde{x}_i$ by setting $\tilde{x}_i^{(\ell)} = \sqrt{L_\ell} x_i^{(\ell)}$ and $\tilde{x}_*^{(\ell)} = \sqrt{L_\ell} x_*^{(\ell)}$ for all $\ell \in \{1, \dots, p\}$. Since x_* is a minimizer of f , $\tilde{x}_*$ is a minimizer of $\tilde{f}$, and it is easy to verify that

$$\|\tilde{x}_0 - \tilde{x}_*\|_{(1, \dots, 1)} = \|x_0 - x_*\|_{\mathbf{L}}.$$

Since the iterates x_i satisfy

$$x_i = x_{i-1} - \frac{\gamma_\ell}{L_\ell} U_\ell \nabla^{(\ell)} f(x_{i-1}),$$

it follows that x_i and x_{i-1} only differ in their ℓ^{th} block of coordinates and thus $\tilde{x}_i$ and $\tilde{x}_{i-1}$ also only differ in their ℓ^{th} block of coordinates and

$$\begin{aligned} \tilde{x}_i - \tilde{x}_{i-1} &= \tilde{x}_i^{(\ell)} - \tilde{x}_{i-1}^{(\ell)} \\ &= \sqrt{L_\ell} (x_i^{(\ell)} - x_{i-1}^{(\ell)}) \\ &= -\sqrt{L_\ell} \left(\frac{\gamma_\ell}{L_\ell} U_\ell \nabla^{(\ell)} f(x_{i-1}) \right) \\ &= -\frac{\gamma_\ell}{\sqrt{L_\ell}} U_\ell \nabla^{(\ell)} f(x_{i-1}). \end{aligned}$$

Using the fact that

$$\nabla^{(\ell)} \tilde{f}(\tilde{x}_{i-1}) = \frac{1}{\sqrt{L_\ell}} \nabla^{(\ell)} f(x_{i-1}),$$

we obtain

$$\tilde{x}_i = \tilde{x}_{i-1} - \gamma_\ell U_\ell \nabla^{(\ell)} \tilde{f}(\tilde{x}_{i-1}),$$

which corresponds to the iterates of $(\text{CCD})(\gamma_\ell)$ applied to $\tilde{f}$. Moreover, it is straightforward to verify that

$$\tilde{f}(\tilde{x}_{pK}) - \tilde{f}(\tilde{x}_*) = f(x_{pK}) - f(x_*).$$

Thus, for any function $f \in \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, we have constructed a function $\tilde{f} \in \mathcal{F}_{0,(1,\dots,1)}^{\text{coord}}(\mathbb{R}^d)$ such that $(\text{CCD})(\frac{\gamma_\ell}{L_\ell})$ applied to $\tilde{f}$ has the same performance as $(\text{CCD})(\gamma_\ell)$ applied to f . Conversely, for any function $f \in \mathcal{F}_{0,(1,\dots,1)}^{\text{coord}}(\mathbb{R}^d)$, using a similar reasoning, we can show that for the function $\tilde{f} : x \mapsto f(\sqrt{L_1}x^{(1)}, \dots, \sqrt{L_p}x^{(p)})$, K cycles of $(\text{CCD})(\frac{\gamma_\ell}{L_\ell})$ achieve the same performance on $\tilde{f}$ as K cycles of $(\text{CCD})(\gamma_\ell)$ on f . This finally gives us

$$\mathcal{W}_{\mathbf{L}}^{\text{CCD}(\frac{\gamma_\ell}{L_\ell})}(p, K) = \mathcal{W}_{(1,\dots,1)}^{\text{CCD}(\gamma_\ell)}(p, K),$$

which concludes the proof. $\square$

Remark 3.26. Note that, thanks to Theorem 3.25, when using the initial condition $\|x_0 - x_*\|_{\mathbf{L}}^2 \leq R_i^2$ in the PEP framework (Setting INIT.), we can, without loss of generality, restrict ourselves to $\mathbf{L} = (1, \dots, 1)$ and $R_i = 1$.

Theorem 3.27. *Given a number of blocks p and a vector of nonnegative constants $\mathbf{L} = (L_1, \dots, L_p)$, the worst-case performance of (CCD) with step sizes $\{\frac{\gamma_\ell}{L_\ell}\}_{\ell=1}^p$ over the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}$ is at least p times worse than the worst-case after pK steps of full gradient descent (MGD) with step sizes $\alpha_i = \gamma_{i \bmod p+1}$ over the class $\mathcal{F}_{0,1}$:*

$$\mathcal{W}_{\mathbf{L}}^{\text{CCD}(\frac{\gamma_\ell}{L_\ell})}(K, p) \geq p \mathcal{W}_1^{\text{MGD}(\gamma_{i \bmod p+1})}(pK)$$

Proof. The idea of the proof is to construct a function $\hat{f} \in \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}$ such that the performance of $\text{CCD}(\frac{\gamma_\ell}{L_\ell})$ on $\hat{f}$ is exactly p times the worst-case performance of $\text{MGD}(\gamma_{i \bmod p})$ over the class $\mathcal{F}_{0,1}$. Thanks to the scale invariance property in Theorem 3.25, we have:

$$\mathcal{W}_{\mathbf{L}}^{\text{CCD}(\frac{\gamma_\ell}{L_\ell})}(p, K) = \mathcal{W}_{(1,\dots,1)}^{\text{CCD}(\gamma_\ell)}(p, K).$$

Thus, it suffices to prove that

$$\mathcal{W}_{(1,\dots,1)}^{\text{CCD}(\gamma_\ell)}(p, K) \geq p \mathcal{W}_1^{\text{MGD}(\gamma_{i \bmod p+1})}(pK).$$

Consider a function $f \in \mathcal{F}_{0,1}$ on which the algorithm (MGD)($\gamma_{i \bmod p+1}$) attains its worst-case performance, and let d be its dimension. Denote by x_* a minimizer of f , and choose $x_0 \in \mathbb{R}^d$ such that $\|x_0 - x_*\|_1^2 \leq R^2$. Now, consider a block-coordinate decomposition of $\mathbb{R}^{pd}$ such that we can write $x = (x^{(1)}, \dots, x^{(p)})^T \in \mathbb{R}^{pd}$, where each block of coordinates is of size d . Define the function $\hat{f}$ by

$$\hat{f}(x) = f(x^{(1)} + \dots + x^{(p)}).$$

Since f is a 1-smooth convex function, the function $\hat{f}$ belongs to $\mathcal{F}_{0,(1,\dots,1)}^{\text{coord}}$. For all $x = (x^{(1)}, \dots, x^{(p)})^T \in \mathbb{R}^{pd}$, define $s(x) = \sum_{\ell=1}^p x^{(\ell)}$, so that for all $x \in \mathbb{R}^{pd}$, we have

$$\hat{f}(x) = f(s(x)).$$

Now, set the initial point for (CCD)(γ_ℓ) on $\hat{f}$ as $\hat{x}_0 \in \mathbb{R}^{pd}$ with

$$\hat{x}_0^{(\ell)} = \frac{x_0}{p}, \quad \forall \ell \in \{1, \dots, p\}.$$

Since x_* is a minimizer of f , the point $\hat{x}_* \in \mathbb{R}^{pd}$ defined by

$$\hat{x}_*^{(\ell)} = \frac{x_*}{p}, \quad \forall \ell \in \{1, \dots, p\}$$

is a minimizer of $\hat{f}$. With these definitions, we have $s(x_0) = x_0$, $s(x_*) = x_*$, and

$$\begin{aligned} \|\hat{x}_0 - \hat{x}_*\|_{(1,\dots,1)}^2 &= \sum_{\ell=1}^p \left\| \frac{x_0}{p} - \frac{x_*}{p} \right\|^2 \\ &= \frac{1}{p^2} \sum_{\ell=1}^p \|x_0 - x_*\|^2 = \frac{1}{p} \|x_0 - x_*\|_1^2 \\ &= \frac{1}{p} \|s(x_0) - s(x_*)\|_1^2. \end{aligned}$$

At any iteration i of (CCD)(γ_ℓ), where $\ell = i \bmod p + 1$, the iterates $\hat{x}_i$ satisfy the update rule:

$$\hat{x}_i^{(\ell)} = \hat{x}_{i-1}^{(\ell)} - \gamma_\ell \nabla^{(\ell)} \hat{f}(\hat{x}_{i-1}).$$

This implies that

$$s(x_i) = s(x_{i-1}) - \gamma_\ell \nabla^{(\ell)} \hat{f}(\hat{x}_{i-1}).$$

By definition of $\hat{f}$, we have $\nabla^{(\ell)} \hat{f}(\hat{x}_{i-1}) = \nabla f(s(x_{i-1}))$, so that

$$s(x_i) = s(x_{i-1}) - \gamma_\ell \nabla f(s(x_{i-1})).$$

This shows that $\{s(x_i)\}$ corresponds to the iterates of $(\text{MGD})(\gamma_{i \bmod p+1})$ on f . Since f is the worst-case function for $(\text{MGD})(\gamma_{i \bmod p+1})$, we have that

$$\mathcal{W}_1^{\text{MGD}(\gamma_{i \bmod p})}(pK) = \frac{f(s(x_{pK})) - f(s(x_*))}{\|s(x_0) - s(x_*)\|_1^2}.$$

Since $\hat{f}(\hat{x}_i) = f(s(x_i))$ and $\|\hat{x}_i - \hat{x}_*\|_{(1, \dots, 1)}^2 = \frac{1}{p} \|s(x_0) - s(x_*)\|_1^2$, we deduce that the performance of $(\text{CCD})(\gamma_\ell)$ after K cycles on $\hat{f}$ is given by

$$\frac{\hat{f}(\hat{x}_{pK}) - \hat{f}(\hat{x}_*)}{\|\hat{x}_0 - \hat{x}_*\|_{(1, \dots, 1)}^2} = p \frac{f(s_{pK}) - f(s(x_*))}{\|s(x_0) - s(x_*)\|_1^2} = p \mathcal{W}_1^{\text{MGD}(\gamma_{i \bmod p})}(pK).$$

By the definition of the worst-case performance of $(\text{CCD})(\gamma_\ell)$ over $\mathcal{F}_{0, (1, \dots, 1)}^{\text{coord}}$, we conclude that

$$\mathcal{W}_{(1, \dots, 1)}^{\text{CCD}(\gamma_\ell)}(p, K) \geq p \mathcal{W}_1^{\text{MGD}(\gamma_{i \bmod p+1})}(pK).$$

This completes the proof. $\square$

Remark 3.28. The previous theorem can be generalized to any deterministic strategy for selecting the block of coordinates to update, other than the cyclic order.

Remark 3.29. The upper bound on the convergence of (CCD) derived in [91, Theorem 1] (also using PEP) violates the lower bound established in Theorem 3.27 and thus appears to be incorrect.

We now provide a descent lemma for the 2-block (CCD) and derive from this lemma a bound on the residual gradient norm for the 2-block (CCD) .

Lemma 3.30 (Descent lemma for 2-block (CCD)). *Given two successive iterates $x_0 = (x_0^{(1)}, x_0^{(2)})^T$ and $x_1 = (x_1^{(1)}, x_1^{(2)})^T$ of $(\text{CCD})\left(\frac{1}{L_\ell}\right)$ on a function f that belongs to $\mathcal{F}_{0, L}^{\text{coord}}(\mathbb{R}^d)$, where the update is performed on the block $\ell \in \{1, 2\}$:*

$$x_1 = x_0 - \frac{1}{L_\ell} U_\ell \nabla^{(\ell)} f(x_0), \quad (3.21)$$

we have:

$$f(x_0) - f(x_1) \geq \frac{1}{2L_\ell} \left(\|\nabla^{(\ell)} f(x_0)\|^2 + \|\nabla^{(\ell)} f(x_1)\|^2 \right). \quad (3.22)$$

Proof. Since f belongs to $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, Theorem 3.7 gives:

$$f(x_0) - f(x_1) \geq \langle \nabla f(x_1), x_0 - x_1 \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_0)\|^2.$$

Since x_0 and x_1 differ only in the ℓ -th block of coordinates, we have

$$\langle \nabla f(x_1), x_0 - x_1 \rangle = \langle \nabla^{(\ell)} f(x_1), x_0^{(\ell)} - x_1^{(\ell)} \rangle,$$

leading to:

$$f(x_0) - f(x_1) \geq \langle \nabla^{(\ell)} f(x_1), x_0^{(\ell)} - x_1^{(\ell)} \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_0)\|^2. \quad (3.23)$$

Substituting (3.21) into (3.23) yields

$$f(x_0) - f(x_1) \geq \frac{1}{L_\ell} \langle \nabla^{(\ell)} f(x_1), \nabla^{(\ell)} f(x_0) \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_0)\|^2.$$

Since

$$\begin{aligned} \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_0)\|^2 &= \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1)\|^2 \\ &\quad - \frac{1}{L_\ell} \langle \nabla^{(\ell)} f(x_1), \nabla^{(\ell)} f(x_0) \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_0)\|^2 \end{aligned}$$

we have

$$\begin{aligned} \langle \nabla^{(\ell)} f(x_1), x_0^{(\ell)} - x_1^{(\ell)} \rangle + \frac{1}{2L_\ell} \|\nabla^{(\ell)} f(x_1) - \nabla^{(\ell)} f(x_0)\|^2 &= \\ \frac{1}{2L_\ell} \left(\|\nabla^{(\ell)} f(x_0)\|^2 + \|\nabla^{(\ell)} f(x_1)\|^2 \right). \end{aligned}$$

Plunging this into (3.23) gives us the desired result and concludes the proof. $\square$

Remark 3.31. Lemma 3.30 is analogous to the double sufficient decrease lemma derived in [100] for full gradient descent.

Theorem 3.32. *Given a vector of nonnegative constants L and a function f that*

belongs to $\mathcal{F}_{0,L}^{coord}(\mathbb{R}^d)$, after K cycles of the 2-block (CCD) $\left(\frac{1}{L_\ell}\right)$ applied on f , we have:

$$\min_{i \in [1, 2K-1]} \|\nabla f(x_i)\|_L^{*2} \leq \frac{2}{2K-1} (f(x_0) - f(x_{2K})). \quad (3.24)$$

Proof. Consider one cycle of 2-block coordinate descent. Without loss of generality, we can order the partial gradient updates as follows:

$$x_1 = x_0 - \frac{1}{L_1} U_1 \nabla^{(1)} f(x_0),$$

$$x_2 = x_1 - \frac{1}{L_2} U_2 \nabla^{(2)} f(x_1).$$

By Lemma (3.30), we have:

$$f(x_0) - f(x_1) \geq \frac{1}{2L_1} \left(\|\nabla^{(1)} f(x_1)\|^2 + \|\nabla^{(1)} f(x_0)\|^2 \right), \quad (3.25)$$

and

$$f(x_1) - f(x_2) \geq \frac{1}{2L_2} \left(\|\nabla^{(2)} f(x_1)\|^2 + \|\nabla^{(2)} f(x_2)\|^2 \right). \quad (3.26)$$

Summing the two inequalities gives:

$$\begin{aligned} f(x_0) - f(x_2) &\geq \frac{1}{2L_1} \|\nabla^{(1)} f(x_0)\|^2 + \frac{1}{2} \|\nabla f(x_1)\|_L^{*2} + \frac{1}{2L_2} \|\nabla^{(2)} f(x_2)\|^2 \\ &\geq \frac{1}{2} \|\nabla f(x_1)\|_L^{*2}, \end{aligned}$$

which corresponds exactly to (3.24) for one cycle ($K = 1$). The proof easily generalizes to K cycles. By writing all the valid inequalities (3.22) between consecutive iterates and summing them, we obtain:

$$\begin{aligned} f(x_0) - f(x_{2K}) &\geq \frac{1}{2L_1} \|\nabla^{(1)} f(x_0)\|^2 + \frac{1}{2} \sum_{i=1}^{2K-1} \|\nabla f(x_i)\|_L^{*2} \\ &\quad + \frac{1}{2L_2} \|\nabla^{(2)} f(x_{2K})\|^2. \end{aligned}$$

Since:

$$\frac{1}{2L_1} \|\nabla^{(1)} f(x_0)\|^2 + \frac{1}{2} \sum_{i=1}^{2K-1} \|\nabla f(x_i)\|_{\mathbf{L}}^{*2} + \frac{1}{2L_2} \|\nabla^{(2)} f(x_{2K})\|^2 \geq \frac{2K-1}{2} \min_{i \in [1, 2K-1]} \|\nabla f(x_i)\|_{\mathbf{L}}^{*2},$$

we obtain the desired result. $\square$

Note that the previous theorem is not trivial to generalize in an analytic form for a number of blocks $p > 2$. We conclude this section by discussing how descent lemmas can be generalized beyond the 2-block (CCD) case and how semi-analytical upper bounds on the convergence of (CCD) can be obtained using Performance Estimation Problems to generate valid descent lemmas.

Theorem 3.33. *Given a number of blocks p and a vector of nonnegative constants L , let $\{x_i\}_{i=\{1, \dots, N\}}$, $N = pK$, be the sequence of iterates generated by the p -block (CCD) $(\frac{1}{L_\ell})$ over a function f that belongs to $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. If we assume that there exists a nonnegative constant C such that for all $k \in \{1, \dots, K-1\}$:*

$$f(x_{(k+1)p}) - f(x_{kp}) \geq C \|\nabla f(x_{kp})\|^2, \quad (3.27)$$

then for all $k \in \{1, \dots, K\}$

$$f(x_{kp}) - f(x_*) \leq \frac{1}{C} \frac{R^2}{k+m}, \quad (3.28)$$

with $R = \min_{x_* \in X_*} \max_{k \in \{1, \dots, K\}} \|x_{pk} - x_*\|$ and $m = \frac{2}{pL_{\max}C}$ where $L_{\max} = \max_{\ell \in \{1, \dots, p\}} L_\ell$.

Proof. As explained in Section 3.4 (when presenting Setting ALL) this is adapted from [14, Corollary 3.7]. Their proof uses the inequalities:

$$\forall k \in 1, \dots, K, \quad \|x_{pk} - x_*\| \leq \max_{x_* \in X_*} \max_{x \in \mathbb{R}^d} \{\|x - x_*\| : f(x) \leq f(x_0)\}.$$

But remains valid, considering the optimal point x_* such that

$$\forall k \in 1, \dots, K, \quad \|x_{pk} - x_*\| \leq \min_{x_* \in X_*} \max_{k \in 1, \dots, K} \|x_{pk} - x_*\|,$$

and using these inequalities instead. Furthermore, the proof in [14, Corol-

lary 3.7] is based on the following descent lemma

$$f(x_{(k+1)p}) - f(x_{kp}) \geq \frac{1}{4L_{\max}(1 + p \frac{L^2}{L_{\min}^2})} \|\nabla f(x_{kp})\|^2, \quad (3.29)$$

which can be replaced by the descent lemma (3.27) giving the desired upper bound (3.28). $\square$

The main challenge here is to find a constant C such that the descent property in (3.27) is satisfied. Unlike the simpler case of only two blocks of coordinates, it is difficult to analytically determine a better constant than the one used in descent lemma (3.29). However, we can numerically obtain an optimized constant by solving the following Performance Estimation Problem:

$$\begin{aligned} C_{\text{opt}} &= \min_{\mathcal{S}_p = \{(x_i, g_i, f_i)\}_{i \in \{1, \dots, p\}}} f(x_0) - f(x_p), \\ &\text{such that } \mathcal{S}_p \text{ is } \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)\text{-interpolable,} \quad (\text{PEP-coord-descent}) \\ &x_1, \dots, x_p \text{ are generated from } x_0 \text{ by (CCD),} \\ &\|\nabla f(x_0)\|^2 = 1. \end{aligned}$$

By definition, it is straightforward to see that C_{opt} is the optimal constant for which (3.27) remains valid. Since (CCD) is a fixed-step first-order BCD algorithm, Problem (PEP-coord-descent) can be relaxed into a convex SDP program, following a similar construction to the one presented in Section 3.5. This provides a computationally efficient way to compute optimized constants ensuring the validity of (3.27), which, by Theorem 3.33, leads to a semi-analytical upper bound on the convergence of (CCD).

Remark 3.34. Note that the approach for automatically designing a descent lemma for (CCD) can be directly adapted to any deterministic fixed-step first-order BCD method for which a convex PEP formulation exists. Moreover, the standard PEP approach for fixed-step first-order BCD algorithms, as presented in Section 3.5, does not generally produce bounds that are easily identifiable analytically. Using PEP to establish descent lemmas, as described above, is a viable method for deriving semi-analytical convergence bounds with simple expressions.

Furthermore, computing optimized descent lemmas is computationally more efficient, since establishing a descent lemma with PEP only requires

solving an SDP involving p matrices of size $(p + 2) \times (p + 2)$, where p is the number of blocks. However, compared to solving the standard PEP described in Section 3.5 which guarantees the optimality of the upper bound with respect to the set of interpolation constraints used, the descent lemma approach yields more conservative upper bounds on the worst-case convergence of BCD algorithms.

In conclusion, PEP-assisted design of optimized descent lemmas offers a useful trade-off between simplicity and accuracy in obtaining upper bounds on the convergence rate of BCD algorithms.

3.7 Numerical Experiments on the Convergence of Block Coordinate-wise Algorithms on the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$

We now provide tighter numerical bounds on the convergence of the three BCD algorithms defined in Section 3.4: cyclic coordinate descent (CCD), alternating minimization (AM), and cyclic accelerated coordinate descent (CACD). As explained in Section 3.4, the standard initial assumption for the analysis of BCD algorithms is that the set $S = \{x \in \mathbb{R}^d : f(x) \leq f(x_0)\}$ is compact. This implies that

$$\|x_{pk} - x_*\| \leq R(x_0), \quad \forall k \in \{1, \dots, K\},$$

where $R(x_0)$ is defined as

$$R(x_0) = \max_{x_* \in X_*} \max_{x \in \mathbb{R}^d} \{\|x - x_*\| : f(x) \leq f(x_0)\}.$$

We then introduced Setting ALL with the conditions (3.12), which is an adaptation of this assumption to our PEP framework. This ensures that the existing upper bounds on the convergence of these algorithms remain valid under the assumptions of Setting ALL. Let us now review the existing convergence bounds for the three BCD algorithms under consideration over the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. For cyclic coordinate descent, we have the following upper bound:

Theorem 3.35. [14, Corollary 3.7] *Given a number of blocks of coordinates p , a vector of nonnegative constant L and a function $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, let $\{x_i\}_{i \in \{1, \dots, N\}}$, $N = pK$, be the sequence of iterates generated by K cycles of p -block cyclic coordinate descent with step-sizes $\gamma_\ell = \frac{1}{L_\ell}$, $\forall \ell \in \{1, \dots, p\}$, (CCD) $\left(\frac{1}{L_\ell}\right)$, then*

it holds that:

$$f(x_{pK}) - f(x_*) \leq 4L_{\max} \left(1 + p^3 \frac{L_{\max}^2}{L_{\min}^2} \right) \frac{1}{K + \frac{8}{p}} R_a^2.$$

where $L_{\max} = \max_{\ell \in 1, \dots, p} L_\ell$ and $L_{\min} = \min_{\ell \in 1, \dots, p} L_\ell$.

Proof. This is adapted from [14, Corollary 3.7]. The only difference is that their proof uses the inequalities:

$$\forall k \in 1 \dots K, \|x_{pk} - x_*\| \leq \max_{x_* \in X_*} \max_{x \in \mathbb{R}^d} \{\|x - x_*\| : f(x) \leq f(x_0)\}.$$

We see that it remains valid considering the optimal point x_* such that

$$\forall k \in 1 \dots K, \|x_{pk} - x_*\| \leq \min_{x_* \in X_*} \max_{k \in 1, \dots, K} \|x_{pk} - x_*\|,$$

and using these inequalities instead. □

In [52], the authors provide an upper bound for (CCD) of similar order to that of the previous theorem, while also covering the non-smooth case. An improved bound is derived in [93], but under more restrictive assumptions, namely considering only scalar blocks of coordinates and functions that admit second-order derivatives.

Similarly, we can adapt the bound obtained in [14] for the 2-block alternating minimization algorithm to hold under Setting ALL:

Theorem 3.36. [14, Theorem 5.2] *Given a vector of nonnegative constants $\mathbf{L} = (L_1, L_2)$ and a function $f \in \mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, let $\{x_i\}_{i \in \{1, \dots, N\}}$, $N = 2K$, the sequence of iterates generated by the K cycles of 2-block (AM) on f then it holds that:*

$$f(x_{2K}) - f(x_*) \leq \frac{2 \min\{L_1, L_2\} R_a^2}{K - 1}$$

For the same reasons stated in Section 3.4, we introduce Setting INIT for the analysis of BCD algorithms over the class $\mathcal{F}_{0, \mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. This setting differs slightly from the one introduced for the class $\mathcal{F}_{0, \mathbf{L}}(\mathbb{R}^d)$:

Setting INIT. Given the starting point x_0 of the BCD algorithm and an

optimal point x_* of the function, we assume that

$$\|x_0 - x_*\|_{\mathbf{L}}^2 \leq R_i^2, \quad (3.30)$$

where $\|\cdot\|_{\mathbf{L}}^2$ is the norm defined in (3.2).

Under this setting, the algorithm (RACD) achieves an accelerated convergence rate over the functional class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$:

Theorem 3.37. [36, Theorem 3] *Given a number of blocks p , a vector of nonnegative constants $\mathbf{L}$ and a function $f \in \mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, let $\{x_i\}_{i \in \{1, \dots, N\}}$ be the iterates generated by N steps of p -block of random accelerated coordinate descent with step-sizes $\gamma_\ell = \frac{1}{L_\ell}$, $\forall \ell \in \{1, \dots, p\}$, (RACD) $\left(\frac{1}{L_\ell}\right)$, on f , then it holds that:*

$$\mathbb{E}[f(x_N) - f(x_*)] \leq \frac{4p^2}{(N-1+2p)^2} R^2$$

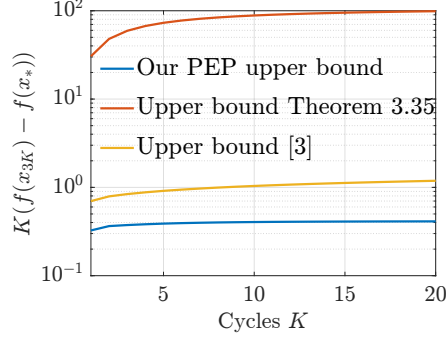
with $R^2 = \left(1 - \frac{1}{p}\right) (f(x_0) - f(x_*)) + \frac{1}{2} \|x_0 - x_*\|_{\mathbf{L}}^2$.

We now present numerical results using the PEP framework derived in Section 3.5 for the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ and compare them with the analytical bounds discussed above.

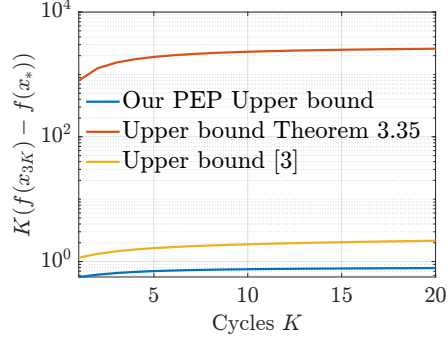
First, we consider the algorithm (CCD) (γ_ℓ) with step sizes $\gamma_\ell = \frac{1}{L_\ell}$ for all $\ell \in \{1, \dots, p\}$. We provide bounds for cases with 2 and 3 blocks and a range of Lipschitz constants. Although we present results only for 2- and 3-block algorithms, our PEP framework can accommodate an arbitrary number of blocks of coordinates. As a performance criterion, we measure the difference between the objective value at the final iterate and the optimal value, $f(x_{pK}) - f(x_*)$. Since a sublinear convergence rate of $\mathcal{O}(1/K)$ is expected, we present our results in the form of the bound multiplied by the number of cycles, to facilitate analysis.

Figures 3.6 and 3.7 show that the obtained upper bounds outperform the best-known analytical bound given in Theorem 3.35 by approximately a multiplicative factor of 10. Additionally, these figures illustrate that our upper bounds improve upon those obtained using the PEP framework from [3] (where the inequalities 2 in Theorem 3.7 are used as interpolation conditions), demonstrating that our framework provides a tighter analysis due to a better characterization (interpolation conditions) of the functional

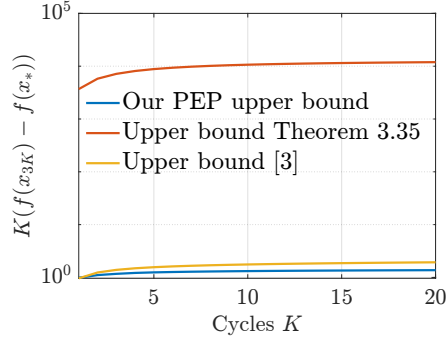
class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$.



(a) $\mathbf{L} = (1, 1, 1)$

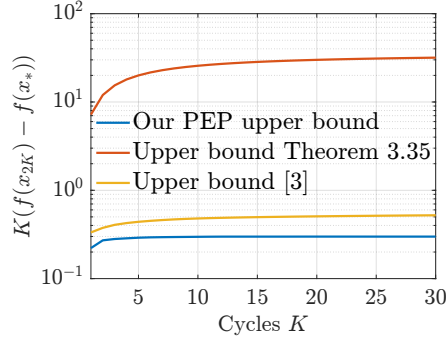


(b) $\mathbf{L} = (1, 1, 3)$

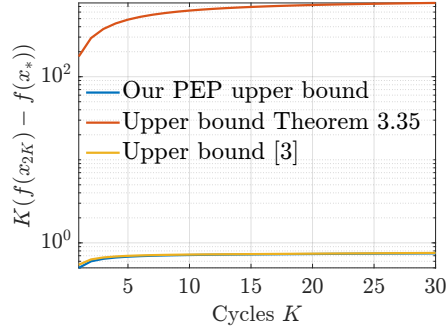


(c) $\mathbf{L} = (1, 3, 5)$

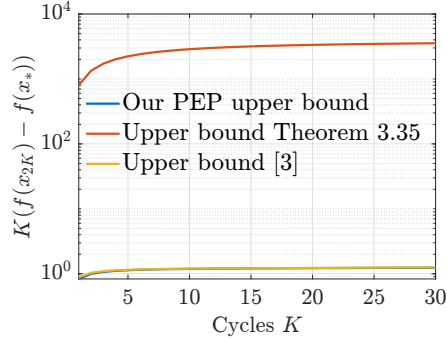
Fig. 3.6 Comparison of upper bounds on the worst-case for 3-block (CCD) in the Setting ALL.



(a) $L = (1, 1)$



(b) $L = (1, 3)$



(c) $L = (1, 5)$

Fig. 3.7 Comparison of upper bounds on the worst-case for 2-block (CCD) in the Setting ALL.

Remark 3.38. Note that the numerical upper bounds we present in this section do not match any simple analytical expressions, which makes it difficult to establish improved analytical upper bounds on the worst-case

convergence rate of BCD algorithms using our PEP framework. We explored several approaches to address this issue, such as removing inactive constraints (i.e., those with zero dual variables), attempting to reconstruct the worst-case function analytically from the PEP outputs, which can be done using a Cholesky factorization of the Gram matrices $G^{(\ell)}$ to retrieve the numerical values of the iterates, gradients, and function values. We also applied several heuristics to find lower-rank solutions to simplify the identification of the bounds. These attempts did not yield satisfactory results.

Furthermore, the difficulty in identifying our upper bounds analytically may stem from the fact that our interpolation conditions are only necessary, and thus characterize a larger set of functions, which might include functions on which the worst-case of BCD algorithms does not admit a simple closed-form expression. It is also possible that the class of coordinate-wise smooth convex functions is not well suited to the analysis of these algorithms. A different function class—still covering essentially the same practical problems, for instance by defining coordinate-wise smoothness differently—might lead to upper bounds that admit simple analytical expressions.

In Figure 3.8, we provide numerical upper bounds for the 2-block $(\text{CCD})\left(\frac{1}{L_\ell}\right)$ and 3-block $(\text{CCD})\left(\frac{1}{L_\ell}\right)$ under the milder assumptions of Setting INIT (see Section 3.4 for a discussion on the two settings). By selecting step-sizes $\gamma_\ell = \frac{1}{L_\ell}$ for $\ell \in \{1, \dots, p\}$, we can assume without loss of generality that $\mathbf{L} = (1, \dots, 1)$ and $R_i = 1$ due to the scaling law stated in Theorem 3.25. We plot lower and upper bounds on the function accuracy after K cycles, $f(x_{pK}) - f(x_*)$, times the number of cycles K . We compare our upper bound to the one obtained using the PEP framework developed in [3], as well as to the lower bound derived in Theorem 3.27. As expected, the upper bound on the worst-case obtained using our PEP framework, outperforms the upper bound obtained using the PEP framework from [3] which shows that we have tighter interpolation conditions.

In Figure 3.9, we provide a linear fit of the reciprocal of our bounds, suggesting that the convergence rate of (CCD) under the assumptions of Setting INIT is $\mathcal{O}(1/K)$. We conclude that the stronger assumptions of Setting ALL do not appear to yield faster convergence compared to the milder assumptions of Setting INIT. To the best of our knowledge, no analytical convergence bound exists for deterministic BCD algorithms under the

milder assumptions of Setting INIT. We thus provide the first numerical evidence suggesting convergence of (CCD) in this setting, and with order $\mathcal{O}(1/K)$.

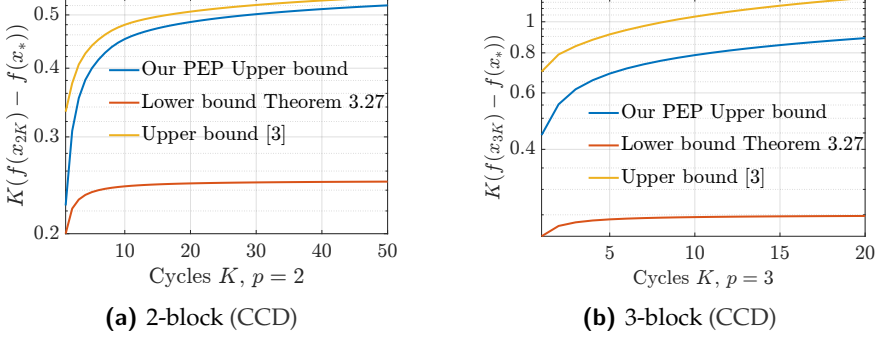


Fig. 3.8 Lower and upper bounds on the worst-case for 2-block and 3-block (CCD) in the Setting INIT.

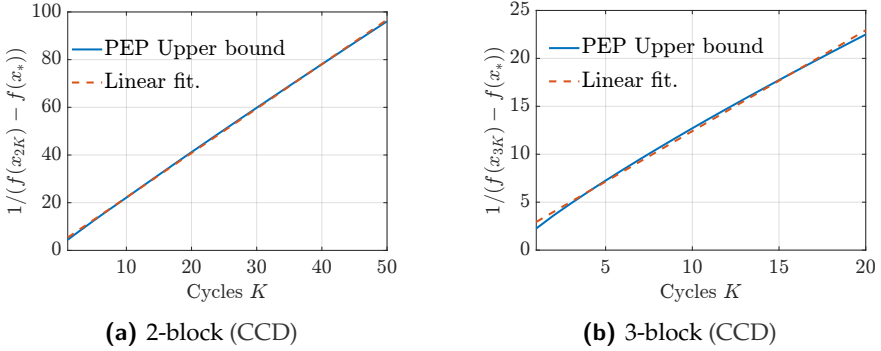


Fig. 3.9 Numerical linear fit of our PEP upper bounds on the worst-case for 2-block and 3-block (CCD) in the Setting INIT.

Conjecture 3.39. *Algorithm (CCD) with step sizes $\frac{1}{L_\ell}$, $\forall \ell \in \{1, \dots, p\}$ has an $\mathcal{O}(1/N)$ convergence rate over the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ in Setting INIT, i.e., after N iterations of (CCD) over a function $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, we have*

$$\frac{f(x_N) - f(x_*)}{\|x_0 - x_*\|_L^2} = \mathcal{O}(1/N).$$

Finding an analytical expression for the numerical upper bounds on the

convergence of (CCD) is challenging. Therefore, in Figure 3.10, we use Problem (PEP-coord-descent) and Theorem 3.33 to present a semi-analytical bound for the 2-block (CCD) $\left(\frac{1}{L_\ell}\right)$ algorithm under the assumptions of Setting ALL with $\mathbf{L} = (1, 1)$ and $R_a = 1$.

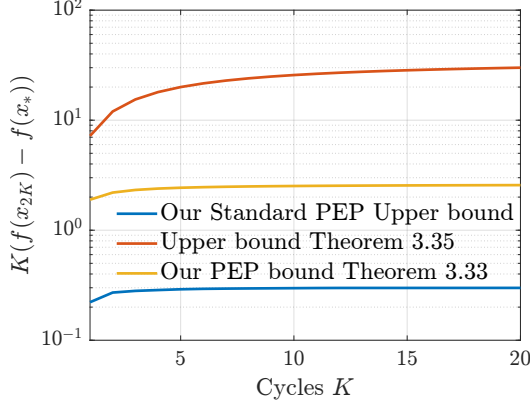


Fig. 3.10 Comparison between the standard PEP bound (blue), the upper bound using PEP to establish a descent lemma (yellow) and the best known analytical bound (red) on the convergence of 2-block (CCD) with $\mathbf{L} = (1, 1)$.

The optimal value of Problem (PEP-coord-descent) in this case is $C_{\text{opt}} = 0.38$. Figure 3.10 shows that the upper bound, obtained using our PEP-based descent lemma, outperforms the bound given in Theorem 3.35, originally derived in [14] using the suboptimal descent Lemma 3.29. As explained in Section 3.5, the standard PEP framework for BCD algorithms over the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$ yields the best possible upper bound given the interpolation constraints characterizing the function class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. Consequently, bounds obtained through a descent lemma constructed using PEP are inherently suboptimal compared to those derived directly via the standard PEP approach, as confirmed in Figure 3.10. However, as illustrated here, the approach based on PEP-generated descent lemma provides a suitable compromise between the tightness and simplicity of the resulting upper bound.

Using the PEP framework, we further analyze certain properties of the convergence of (CCD) $\left(\frac{1}{L_\ell}\right)$ over the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. Specifically, we study the dependence of the worst-case convergence rate of (CCD) with respect to

the number of coordinate blocks p , as well as the optimal constant step size with respect to the number of blocks p and cycles K , under the assumptions of Setting INIT with $\mathbf{L} = (1, 1)$ and $R_i = 1$.

In Figure 3.11, we plot the upper bound on the function accuracy, $f(x_{pK}) - f(x_*)$ obtained using PEP, with respect to the number of blocks p for 1, 2, and 3 cycles. In Figure 3.12, we provide a linear fit for these plots. The numerical results of Figures 3.11 and 3.12 suggest that our upper bound grows linearly with the number of blocks p . This result represents an improvement over the analytical bound in Theorem 3.35, which exhibits a cubic dependence on p under the stronger assumptions of Setting ALL. Given the lower bound on the convergence of $(\text{CCD})\left(\frac{1}{L_\ell}\right)$ derived in Theorem 3.27, these numerical results suggest that, under Setting INIT, a linear dependence of the worst-case convergence rate of $(\text{CCD})\left(\frac{1}{L_\ell}\right)$ on the number of blocks p is optimal.

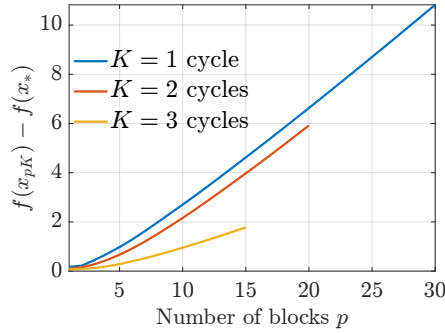


Fig. 3.11 Linear dependency of our PEP upper bound with respect to the number of blocks p for (CCD) in the Setting INIT.

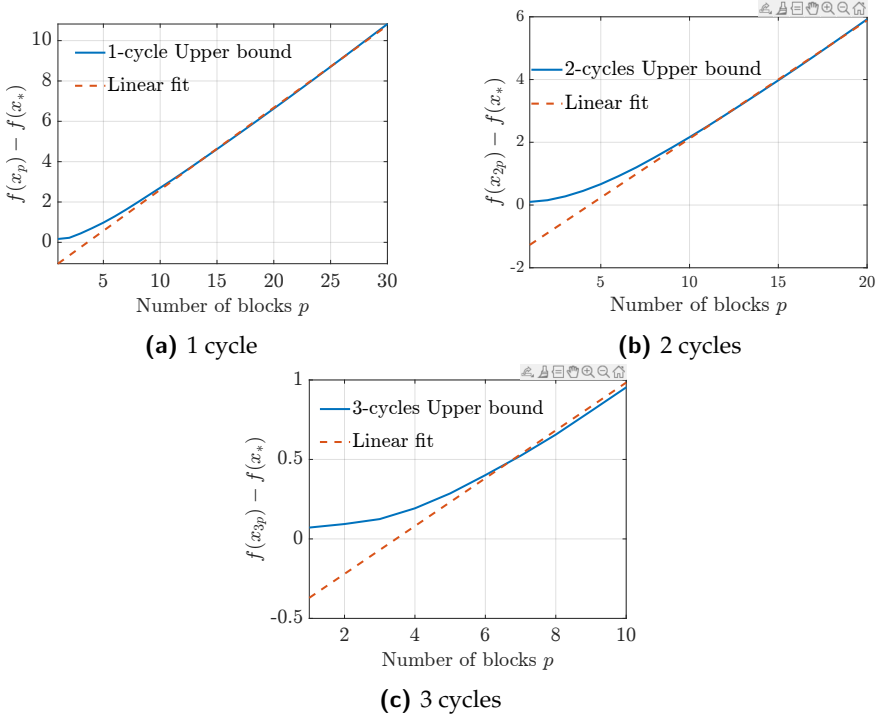


Fig. 3.12 Numerical linear fit of our PEP upper bounds with respect to the number of blocks p for 1 cycle, 2 cycles, and 3 cycles.

Conjecture 3.40. *Under the assumption of Setting INIT, the worst-case convergence rate of (CCD) with step sizes $\frac{1}{L_\ell}$, $\forall \ell \in \{1, \dots, p\}$ grows linearly with respect to the number of blocks p , i.e., after N iterations of (CCD) on a function $f \in \mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, we have*

$$f(x_N) - f(x_*) = p C(L, R, \gamma, N) \|x_0 - x_*\|_{\mathbf{L}}^2,$$

where $C(L, R, \gamma, N)$ is constant with respect to the number of blocks p .

We now utilize PEP to gain insights into the optimal choice of step sizes for (CCD)(γ_ℓ). We specifically focus on step sizes of the form $\gamma_\ell = \frac{\gamma}{L_\ell}$ for each coordinate block $\ell \in \{1, \dots, p\}$. Under Setting INIT, we can assume without loss of generality that $\mathbf{L} = (1, \dots, 1)$ and $R_i = 1$, thanks to the scaling law provided in Theorem 3.25.

Figure 3.13 illustrates the evolution of the PEP upper bound as a function of

the step size γ for the 2-block, 3-block, and 4-block variants of $(\text{CCD})\left(\frac{\gamma}{L_\ell}\right)$ after one cycle ($K = 1$) and three cycles ($K = 3$). Additionally, Table 3.3 provides approximate numerical values of the optimal step sizes minimizing the PEP upper bound for these cases.

We observe that the optimal step size decreases as either the number of blocks or the number of cycles increases. This observation may be explained by the fact that increasing the number of coordinate blocks p or cycles K leads to a higher total number of partial gradient updates, thereby making shorter step sizes more appropriate. Furthermore, we note that the optimal step sizes identified through our numerical experiments are consistently smaller ($\gamma_\ell \leq 1$) than the conjectured optimal step sizes for (MGD) discussed in [33, 97]. Unfortunately, deriving analytical expressions for these optimal step sizes currently appears out of reach.

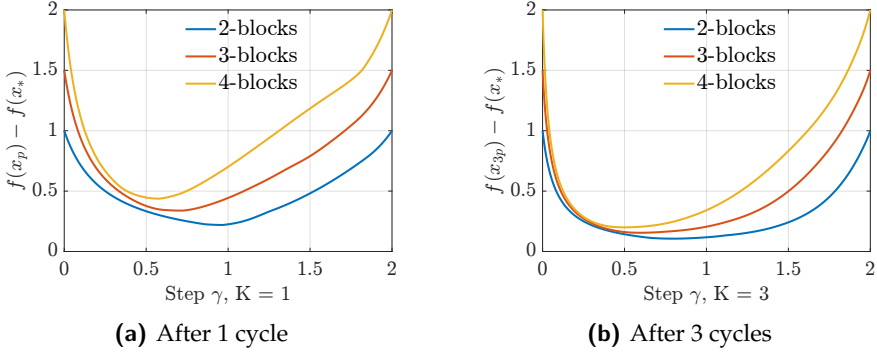


Fig. 3.13 PEP upper-bound on the convergence of $\text{CCD}\left(\frac{\gamma}{L_\ell}\right)$ in the Setting INIT with respect to the step-size γ for 2, 3, and 4 blocks of coordinates after 1 and 3 cycles.

Number of blocks p	Optimal step size ($K = 1$)	Optimal step size ($K = 3$)
2	0.967	0.796
3	0.700	0.596
4	0.576	0.496

Table 3.3 Optimal choice of step size γ for 1 and 3 cycles of 2-block, 3-block and 4-block $(\text{CCD})\left(\frac{\gamma}{L_\ell}\right)$

Conjecture 3.41. *Under the assumptions of setting INIT, the optimal constant*

step sizes for CCD are shorter than $\frac{1}{L_\ell}$, $\ell \in \{1, \dots, p\}$.

We now turn our attention to the alternating minimization algorithm. Specifically, we analyze the 2-block alternating minimization algorithm (AM) under Setting ALL for various choices of Lipschitz constants with $R_a = 1$. Although our framework accommodates an arbitrary number of blocks, we restrict our analysis to the 2-block case to facilitate comparison between our numerical upper bounds and the analytical bound previously established in [14] (see Theorem 3.36). Figure 3.14 indicates that the existing bound for the 2-block (AM) can be improved by approximately a multiplicative factor of 2. This numerical result confirms the observations regarding the convergence of (AM) presented in Section 3.4, obtained using the PEP framework for the function class $\mathcal{F}_{0,L}(\mathbb{R}^d)$.

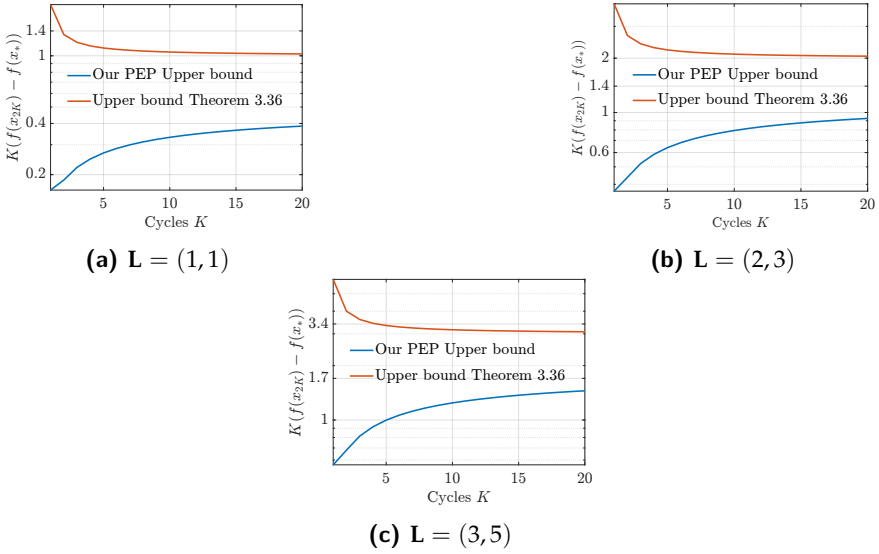


Fig. 3.14 Comparison of upper bounds on the worst-case for 2-block (AM) for $L = (1, 1)$, $L = (2, 3)$, and $L = (3, 5)$ in the Setting ALL.

We conclude this section with an analysis of (CACD) over the function class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, comparing it to its randomized counterpart (RACD). Recall that (RACD) achieves an accelerated convergence rate of $\mathcal{O}(1/N^2)$ on $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$, where N denotes the number of iterations. In Section 3.4, we utilized the PEP framework for BCD algorithms over the class $\mathcal{F}_{0,L}(\mathbb{R}^d)$ to provide numerical evidence suggesting that (CACD) shows an unacceler-

ated convergence rate of $\mathcal{O}(1/K)$ over this class, with K representing the number of cycles. Here, using our PEP framework adapted for the actual function class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, we present additional numerical experiments supporting the observation that (CACD) with step sizes $\gamma_\ell = \frac{1}{L_\ell}$, for all $\ell \in \{1, \dots, p\}$, exhibits a slower convergence rate compared to its randomized variant.

We place ourselves in Setting INIT. As previously discussed, this allows us to assume without loss of generality that $\mathbf{L} = (1, \dots, 1)$ and $R_i = 1$ due to the scaling law provided by Theorem 3.25. In Figure 3.15, we present valid upper bounds on the worst-case performance of 2-block and 3-block (CACD) $\left(\frac{1}{L_\ell}\right)$, multiplied by the squared number of cycles K^2 . This quantity grows with respect to K and appears to be linear in K after the first few cycles, suggesting a convergence rate slower than $\mathcal{O}(1/K^2)$.

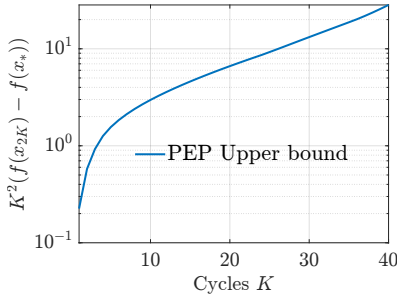
To further investigate this gap between (CACD) and its random variant (RACD), we adopt a similar approach as in Section 3.4, Table 3.1. Table 3.4 reports PEP upper bounds for all possible sequences of coordinate-block updates after $K = 2$ cycles of 2-block (CACD) $\left(\frac{1}{L_\ell}\right)$. Additionally, it provides an upper bound on the worst-case expected performance after $N = 4$ steps of the randomized variant (RACD) $\left(\frac{1}{L_\ell}\right)$, denoted by $\mathcal{W}^{\text{RACD}}$. The bound $\mathcal{W}^{\text{RACD}}$ is computed using the random BCD version of the PEP framework described in Problem (PEP-rdn-coord).

It is noteworthy that $\mathcal{W}^{\text{RACD}}$ is smaller than all the upper bounds for every possible deterministic sequence of updates. This highlights the fact that randomness seems to play a significant role in acceleration for BCD algorithms over the class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$. Intuitively, it appears that for each fixed sequence of coordinate block updates, one can construct a specific worst-case function on which (CACD) $\left(\frac{1}{L_\ell}\right)$ performs poorly. However, it seems that no single function leads to poor performance across all update sequences and thus on average for (RACD) $\left(\frac{1}{L_\ell}\right)$.

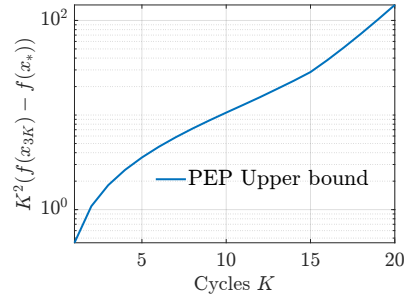
Block choice type	Ordered block choices	Worst-case
Cyclic ($= \mathcal{W}^{\text{CACD}}$)	1 2 1 2 or 2 1 2 1	0.14429
Fixed choice	1 2 2 1 or 2 1 1 2	0.14988
Fixed choice	1 2 1 1 or 2 1 2 2	0.16453
Fixed choice	1 1 2 1 or 2 2 1 2	0.19574
Fixed choice	1 2 2 2 or 2 1 1 1	0.19905
Fixed choice	1 1 2 2 or 2 2 1 1	0.23462
Fixed choice	1 1 1 2 or 2 2 2 1	0.25517
Fixed choice	1 1 1 1 or 2 2 2 2	0.500
Random ($= \mathcal{W}^{\text{RACD}}$)	at each step: pick 1 or 2	0.1046

Table 3.4 Upper bounds on the convergence rate after 4 iterations of $(\text{CACD})\left(\frac{1}{L_\ell}\right)$ for every possible sequence of block coordinate updates and upper bound on the worst-case expected performance of $(\text{RACD})\left(\frac{1}{L_\ell}\right)$, denoted by $\mathcal{W}^{\text{RACD}}$, obtained via the PEP formulation (PEP-rdn-coord).

Remark 3.42. The results presented in Table 3.4 also tends to confirm the observation made in Section 3.4 that the best deterministic choice of steps is the cyclic one.



(a) 2-block CACD



(b) 3-block CACD

Fig. 3.15 PEP upper bound multiplied by the squared number of cycles K^2 for 2-block (left) and 3-block (right) $(\text{CACD})\left(\frac{1}{L_\ell}\right)$, indicating convergence slower than $\mathcal{O}\left(\frac{1}{K^2}\right)$.

Conjecture 3.43. Algorithm (CACD) with step sizes $\frac{1}{L_\ell}$ has a convergence rate of $\mathcal{O}(1/K)$ over the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$ under Setting INIT, i.e., after K cycles with

step sizes $\frac{1}{L_\ell}$, we have

$$\frac{f(x_{pK}) - f(x_*)}{\|x_0 - x_*\|_L^2} = \mathcal{O}(1/K).$$

3.8 Conclusion

In this chapter, we introduced a versatile convex PEP framework for the worst-case analysis of block coordinate descent algorithms. We applied this framework to provide new insights about the behaviour of several BCD algorithms. In particular, we provided improved numerical bounds for cyclic coordinate descent (CCD) and alternating minimization (AM) and provided insights on the evolution of the worst-case of (CCD) with respect to the number of blocks as well as the evolution of the optimal step-size (CCD) with respect to the number of blocks and cycles. Additionally, we compared the worst-case performance of cyclic accelerated coordinate descent (CACD) to its random version and observed a slower convergence rate, indicating the significance of randomness in accelerating block coordinate descent algorithms. We demonstrated the convergence of these algorithms under the simplified constraints of Setting INIT, where only a bound on the initial distance to a minimizer is assumed, as opposed to the typically employed stronger assumptions of Setting ALL, which provide bounds on the distances between all the iterates and a minimizer. Furthermore, we established formally several results concerning the behavior of (CCD), including scale invariance with respect to the block coordinate-wise smoothness constants, an improved descent lemma for 2-block (CCD) with a corresponding upper bound on the convergence for the residual gradient norm, and a lower bound on its worst-case performance for the function value accuracy criterion that is equal to the number of blocks multiplied by the worst-case performance of full gradient descent on smooth convex functions. We believe our PEP framework to be a useful tool to guide future research on block coordinate-wise algorithms for example by helping the tuning of step-sizes.

Open research directions include: analytically identifying our numerical upper bounds with corresponding convergence proofs; deriving necessary and sufficient interpolation conditions for the class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$; developing a computationally more efficient framework for analyzing randomized algorithms; and designing accelerated deterministic BCD algorithms.

4

Performance Estimation of First-Order Algorithms over Hölder-Smooth Convex Functions

4.1 Introduction

In classical convex optimization, a function f is often assumed to be differentiable with an L -Lipschitz continuous gradient. This condition ensures that for all pairs of points $x_1, x_2 \in \mathbb{R}^d$, the gradient does not change “too abruptly,” formally stated as

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq L\|x_1 - x_2\|.$$

This assumption simplifies theoretical analyses and underpins many convergence guarantees for standard methods such as (accelerated) gradient descent and its variants [71]. However, in practice, numerous objective functions arising, for instance, in high-dimensional learning problems or tasks involving nonsmooth regularizers, do not satisfy the strict Lipschitz gradient criterion. Instead, their gradients may exhibit a weaker but still

structured form of continuity—Hölder continuity. Specifically, a gradient is said to be β -Hölder continuous if there exists a constant $L_\beta > 0$ such that

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq L_\beta \|x_1 - x_2\|^\beta, \quad 0 < \beta \leq 1. \quad (4.1)$$

When $\beta = 1$, we recover the standard Lipschitz condition, for which the optimal convergence rate for first-order algorithms is known to be $\mathcal{O}(1/N^2)$, where N is the number of iterations, assuming that the function is convex. When $\beta = 0$, we recover the class of nonsmooth convex functions with bounded difference of subgradients, for which the optimal achievable rate with first-order algorithms is $\mathcal{O}(1/\sqrt{N})$. When $0 < \beta < 1$, a complexity lower bound for this class of problems with a rate $\mathcal{O}\left(1/N^{\frac{1+3\beta}{2}}\right)$ is given in [69], interpolating between the optimal convergence rates for smooth convex functions ($\beta = 1$) and nonsmooth convex Lipschitz functions ($\beta = 0$). The first optimal methods for such problems were developed in [67]. A key advantage of these schemes lies in their ability to automatically adapt to the appropriate level of the smoothness parameter β , given knowledge of certain problem characteristics such as estimates of the Lipschitz constant L_β and the distance to the optimum.

In [27], the minimization of Hölder smooth convex functions is discussed within the framework of (δ, L) -inexact oracles. Indeed, minimizing Hölder smooth convex functions can be framed as minimizing smooth convex functions with an inexact oracle, where the level of inexactness depends on the smoothness parameter β . Such problems can then be addressed using an inexact variant of (FGM). However, this approach also requires prior knowledge of problem characteristics, particularly the parameters β and L_β . To alleviate this limitation, the universal gradient method and its accelerated variant, achieving the optimal convergence rate, were proposed in [73]. These methods automatically adapt to the smoothness parameter β without needing prior knowledge of β or L_β , thanks to a line-search strategy. They were subsequently extended to the stochastic optimization setting in [37]. More recently, line-search-free universal gradient methods were proposed in [59] for the deterministic setting and in [84] for the stochastic setting.

Following the same methodology as in the previous chapters, we develop a convex Performance Estimation framework for fixed-step first-order al-

gorithms over the class of Hölder smooth convex functions. Our objective is to use this convex PEP as a tool to guide the search for a fixed-step first-order algorithm that converges over Hölder smooth convex functions and achieves similar or faster convergence rates than universal gradient methods. Consider the unconstrained optimization problem:

$$\min_{f \in \mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)} f(x),$$

where $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ is the class of L_β -Hölder smooth convex functions defined as follows:

Definition 4.1 (L_β -Hölder smooth convex functions: $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$). Given $\beta \in]0, 1]$ and a nonnegative constant L_β , a differentiable function $f : \mathbb{R}^d \mapsto \mathbb{R}$ belongs to $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ if and only if for all pairs $x_1, x_2 \in \mathbb{R}^d$, the following hold:

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle, \quad (4.2)$$

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq L_\beta \|x_1 - x_2\|^\beta. \quad (4.3)$$

The worst-case performance of the first-order algorithm $\mathcal{M}$, as defined in (1.1), over the function class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$, is given by the following infinite-dimensional PEP:

$$\begin{aligned} w(R, \mathcal{M}, N, \mathcal{P}) &= \sup_{f, x_0, \dots, x_N, x_*} \mathcal{P}(f, x_0, \dots, x_N, x_*), \\ \text{such that } f &\in \mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d), \\ x_* &\text{ is optimal for } f, \\ x_1, \dots, x_N &\text{ is generated from } x_0 \text{ by method } \mathcal{M} \text{ by (1.1),} \\ \mathcal{I}(f, x_0, \dots, x_N, x_*) &\leq R. \end{aligned} \quad (\text{i-PEP-Hölder})$$

where $\mathcal{P}$ is a performance criterion and $\mathcal{I}$ an initial condition ensuring that the worst-case of the algorithm is bounded. For the remainder of this chapter, we choose as a performance criterion

$$\mathcal{P}(f, x_0, \dots, x_N, x_*) = f(x_N) - f(x_*),$$

and as initial condition

$$\mathcal{I}(f, x_0, \dots, x_N, x_*) = \|x_0 - x_*\|.$$

In the same fashion as we explained for the classes of smooth convex functions in Chapter 2 and block coordinate-wise smooth convex functions in Chapter 3, we use the notion of interpolability of a finite set by the class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ to obtain an exact finite reformulation of Problem (i-PEP-Hölder):

Definition 4.2 ($\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolability). Let I be an index set, and consider the set of triples $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$, where $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. The set $\mathcal{S}$ is $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolable if and only if there exists a function $f \in \mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ such that $g_i = \nabla f(x_i)$ and $f(x_i) = f_i$ for all $i \in I$.

Using $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolability, the functional decision variable f can be replaced by a set of decision variables $\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}$, with $I = \{0, \dots, N, *\}$, leading to the following equivalent finite-dimensional formulation of Problem (i-PEP-Hölder):

$$\begin{aligned} w(R, \mathcal{M}, N, \mathcal{P}) = & \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} f_N - f_*, \\ & \text{such that } \mathcal{S}_N \text{ is } \mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)\text{-interpolable,} \\ & x_1, \dots, x_N \text{ is generated from } x_0 \text{ by method } \mathcal{M} \text{ with (1.1),} \\ & g_* = 0, \\ & \|x_0 - x_*\| \leq R. \end{aligned} \tag{f-PEP-Hölder}$$

Organization and Contributions. In Section 4.2, we review necessary and sufficient characterizations for the functional classes $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ and derive necessary $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolation conditions. In Section 4.3, we develop a convex PEP framework to establish valid upper bounds on the worst-case convergence of fixed-step first-order algorithms over the class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$. Finally, in Section 4.4, we review existing worst-case complexity results for first-order methods over Hölder-smooth convex functions and propose a simple fixed-step first-order algorithm whose step sizes interpolate between the constant step sizes of gradient descent for smooth convex functions and the decreasing step sizes of the subgradient method for nonsmooth

convex Lipschitz functions. We provide numerical upper bounds on the convergence rate of our algorithm and show that, for Hölder smoothness parameters β above a certain threshold β_0 (i.e., $\beta_0 < \beta < 1$), our algorithm achieves a faster convergence rate than the universal gradient method.

4.2 Hölder-Smooth Convex Functions

To derive a tractable formulation of the PEP for the class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$, we follow a similar approach to that used in previous chapters for the classes $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ and $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. We first review known characterizations of this class (see [109, 43, 10, 105]) and establish the necessity and sufficiency of these conditions when enforced over the entire space $\mathbb{R}^d$. Restricting these conditions to a finite set yields necessary interpolation conditions. In the next two lemmas, we provide valid upper and lower bounds for L_β -Hölder-smooth convex functions.

Lemma 4.3 (Upper bound for $f \in \mathcal{F}_{L_\beta}^\beta(\mathbb{R}^d)$). [105, Lemma 1]. Let $\beta \in]0, 1]$ and $L_\beta \geq 0$. If $f : \mathbb{R}^d \mapsto \mathbb{R}$, a differentiable function, belongs to $\mathcal{F}_{L_\beta}^\beta(\mathbb{R}^d)$, then for all $x_1, x_2 \in \mathbb{R}^d$, the following inequalities hold:

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle, \quad (4.4)$$

$$f(x_2) - f(x_1) \leq \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{L_\beta}{\beta + 1} \|x_2 - x_1\|^{\beta+1}. \quad (4.5)$$

Lemma 4.4 (Lower bound for $f \in \mathcal{F}_{L_\beta}^\beta(\mathbb{R}^d)$). Let $\beta \in]0, 1]$ and $L_\beta \geq 0$. If $f : \mathbb{R}^d \mapsto \mathbb{R}$, a differentiable function, belongs to $\mathcal{F}_{L_\beta}^\beta(\mathbb{R}^d)$, then for all $x_1, x_2 \in \mathbb{R}^d$, the following inequality holds:

$$f(x_2) - f(x_1) \geq \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{\beta}{(\beta + 1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}}. \quad (4.6)$$

Proof. This proof is adapted from [96, Theorem 2.34], which gives a similar result for smooth ($\beta = 1$) convex functions. By Lemma 4.3, for all $x, \Delta x \in \mathbb{R}^d$, we have:

$$f(x + \Delta x) - f(x) - \langle \nabla f(x), \Delta x \rangle \leq \frac{L_\beta}{\beta + 1} \|\Delta x\|^{\beta+1}.$$

Define the following functions h_x and ℓ for any fixed $x \in \mathbb{R}^d$ and all $\Delta x \in \mathbb{R}^d$:

$$h_x(\Delta x) = f(x + \Delta x) - f(x) - \langle \nabla f(x), \Delta x \rangle, \quad \ell(\Delta x) = \frac{L_\beta}{\beta + 1} \|\Delta x\|^{\beta+1}.$$

The previous inequality becomes:

$$h_x(\Delta x) \leq \ell(\Delta x), \quad \forall x, \Delta x \in \mathbb{R}^d,$$

and we observe that both functions are convex. Taking the Fenchel conjugate with respect to Δx reverses the inequality:

$$h_x^*(\Delta g) \geq \ell^*(\Delta g), \quad \forall x, \Delta g \in \mathbb{R}^d.$$

Straightforward computations show that the Fenchel conjugate of ℓ is:

$$\ell^*(\Delta g) = \frac{\beta}{(\beta + 1)L_\beta^{\frac{1}{\beta}}} \|\Delta g\|^{\frac{\beta+1}{\beta}}, \quad \forall \Delta g \in \mathbb{R}^d.$$

For h_x^* , we compute for any $x \in \mathbb{R}^d$:

$$\begin{aligned} h_x^*(\Delta g) &= \sup_{\Delta x \in \mathbb{R}^d} \langle \Delta g, \Delta x \rangle - h(\Delta x) \\ &= f(x) + \sup_{\Delta x \in \mathbb{R}^d} (\langle \nabla f(x) + \Delta g, \Delta x \rangle - f(x + \Delta x)) \quad (\text{definition of } h) \\ &= f(x) + \sup_{x' \in \mathbb{R}^d} (\langle \nabla f(x) + \Delta g, x' - x \rangle - f(x')) \quad (x' = x + \Delta x) \\ &= f(x) - \langle \nabla f(x) + \Delta g, x \rangle + \sup_{x' \in \mathbb{R}^d} (\langle \nabla f(x) + \Delta g, x' \rangle - f(x')) \\ &= f(x) - \langle \nabla f(x) + \Delta g, x \rangle + f^*(\nabla f(x) + \Delta g) \quad (\text{definition of } f^*) \end{aligned}$$

Injecting the expressions of h_x^* and ℓ^* , the inequality $h_x^*(\Delta g) \geq \ell^*(\Delta g)$ becomes for all $x, \Delta g \in \mathbb{R}^d$:

$$f(x) - \langle \nabla f(x) + \Delta g, x \rangle + f^*(\nabla f(x) + \Delta g) \geq \frac{\beta}{(\beta + 1)L_\beta^{\frac{1}{\beta}}} \|\Delta g\|^{\frac{\beta+1}{\beta}}.$$

For any pair $x_1, x_2 \in \mathbb{R}^d$, choosing $x = x_2$ and $\Delta g = \nabla f(x_1) - \nabla f(x_2)$, we

have:

$$f(x_2) - \langle \nabla f(x_1), x_2 \rangle + f^*(\nabla f(x_1)) \geq \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}}.$$

Using Young's equality (see proof of Lemma 3.6), $f(x_1) + f^*(\nabla f(x_1)) = \langle x_1, \nabla f(x_1) \rangle$, yields:

$$f(x_2) - f(x_1) \geq \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}},$$

which completes the proof. $\square$

We now prove that the lower bound established in Lemma 4.4 implies that the function belongs to a $\mathcal{F}_{0, M_\beta}^\beta(\mathbb{R}^d)$ but with $M_\beta \geq L_\beta$.

Lemma 4.5. *Let $\beta \in]0, 1]$, $L_\beta \geq 0$, and $f : \mathbb{R}^d \mapsto \mathbb{R}$ a differentiable function. If For any $x_1, x_2 \in \mathbb{R}^d$, the following inequality holds*

$$f(x_2) - f(x_1) \geq \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}},$$

then $f \in \mathcal{F}_{0, M_\beta}^\beta(\mathbb{R}^d)$ with $M_\beta = \left(\frac{1+\beta}{2\beta}\right)^\beta L_\beta$

Proof. It is straightforward to verify that for all $x_1, x_2 \in \mathbb{R}^d$:

$$f(x_2) - f(x_1) \geq \langle \nabla f(x_1), x_2 - x_1 \rangle.$$

We now show that for all $x_1, x_2 \in \mathbb{R}^d$:

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq M_\beta \|x_2 - x_1\|^\beta.$$

For any $x_1, x_2 \in \mathbb{R}^d$, we have:

$$f(x_2) - f(x_1) \geq \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}},$$

$$f(x_1) - f(x_2) \geq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}}.$$

Summing these inequalities gives:

$$\frac{2\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}} \leq \langle \nabla f(x_2) - \nabla f(x_1), x_2 - x_1 \rangle.$$

Applying the Cauchy-Schwarz inequality yields:

$$\frac{2\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{\beta+1}{\beta}} \leq \|\nabla f(x_1) - \nabla f(x_2)\| \|x_2 - x_1\|.$$

This simplifies to:

$$\|\nabla f(x_1) - \nabla f(x_2)\|^{\frac{1}{\beta}} \leq \frac{1+\beta}{2\beta} L_\beta^{\frac{1}{\beta}} \|x_2 - x_1\|,$$

and raising both sides to the power β gives:

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq \left(\frac{1+\beta}{2\beta} \right)^\beta L_\beta \|x_2 - x_1\|^\beta.$$

which concludes the proof. $\square$

We now derive necessary interpolation conditions for the class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ using the characterization of Lemma 4.4.

Theorem 4.6. *Let I be an index set, and consider the set of triples $\mathcal{S} = \{(x_i, g_i, f_i)\}_{i \in I}$, where $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. If the set $\mathcal{S}$ is $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolable, then for all $i, j \in I$:*

$$f_i - f_j \geq \langle g_j, x_i - x_j \rangle + \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \|g_i - g_j\|^{\frac{\beta+1}{\beta}}. \quad (4.7)$$

Proof. If $\mathcal{S}$ is $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolable, then there exists a function $f \in \mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ such that $f(x_i) = f_i$ and $\nabla f(x_i) = g_i$ for all $i \in I$. Since f belongs to $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$, it satisfies the condition in Lemma 4.4 for any $x_1, x_2 \in \mathbb{R}^d$, and in particular for any x_i, x_j with $i, j \in I$. This directly yields the necessary interpolation condition (4.7). $\square$

Remark 4.7. Particularizing the conditions from Lemma 4.3 to a finite set also yields necessary interpolation conditions. However, expressing the upper

bound inequality that characterizes the Hölder smoothness constraint in a convex form is not straightforward. Therefore, we opt for the characterization of Lemma 4.4, as it provides a more compact representation of the class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$. This characterization inherently incorporates both convexity and Hölder smoothness within a single inequality, and can be formulated in a convex manner using a power cone constraint. The sufficiency of these interpolation conditions, however, remains an open question. In [10], Azagra et al. prove in a broader setting that the interpolation conditions (4.7) ensure the existence of an interpolating function f belonging to $\mathcal{F}_{0,M_\beta}^\beta(\mathbb{R}^d)$, though with $M_\beta > L_\beta$.

4.3 Convex Performance Estimation Problem

In Chapter 2, Section 2.3, we reviewed the convex formulation of the PEP for fixed-step first-order algorithms over the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$, developed in [97]. In Chapter 3, Section 3.5, we extended the PEP framework to fixed-step first-order block coordinate-wise algorithms over the functional class $\mathcal{F}_{0,L}^{\text{coord}}(\mathbb{R}^d)$. Building on this foundation, we now extend the approach to develop a convex PEP framework for fixed-step first-order methods over the functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$. By replacing the $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolability constraint on the set $\mathcal{S}_N$ in Problem (f-PEP-Hölder) with the interpolation conditions (4.7), we obtain:

$$\begin{aligned}
w_\beta^{(d)}(R, L_\beta, \mathcal{M}, N) = & \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} f_N - f_*, \\
& f_i - f_j \geq \langle g_j, x_i - x_j \rangle + \frac{\beta}{(\beta + 1)L_\beta^{\frac{1}{\beta}}} \|g_i - g_j\|^{\frac{\beta+1}{\beta}}, \\
& x_1, \dots, x_N \text{ is generated from } x_0 \text{ by } \mathcal{M} \text{ with (1.1),} \\
& \{x_*, g_*, f_*\} = \{0, 0, 0\}, \\
& \|x_0 - x_*\|_2 \leq R.
\end{aligned}$$

(d-PEP-Hölder)

We can set, without loss of generality, $x_* = 0$ and $f_* = 0$, since the class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ is invariant under additive shifts in function values and translations in the domain. Because the interpolation conditions (4.7) are only necessary, Problem (d-PEP-Hölder) is a relaxation of Problem (f-PEP-Hölder), and consequently of Problem (i-PEP-Hölder). As a result, it only pro-

vides a valid upper bound on the worst-case convergence of the algorithm $\mathcal{M}$. Moreover, Problem (d-PEP-Hölder) is generally nonconvex in its current form due to the quadratic terms in the interpolation conditions. However, we now show that, with our choice of interpolation conditions (4.7), Problem (d-PEP-Hölder) can be reformulated as a convex conic program for any fixed-step first-order algorithm $\mathcal{M}_\alpha$ defined by (2.8). Let us first define the matrix P by stacking the initial point x_0 and the gradients g_i at the iterates x_i :

$$P = [g_0, g_1, \dots, x_0],$$

and the Gram matrix G as $G = P^\top P$. Next, we define the vectors $h_0^\top = [0 \dots 0 \ 1]$, $h_i \in \mathbb{R}^{N+2}$ for $i \in \{1, \dots, N\}$, and h_* as:

$$h_i^\top = [-\alpha_{i,0} \ -\alpha_{i,1} \ \dots \ -\alpha_{i,i-1} \ 0 \ \dots \ 0 \ 1], \quad h_*^\top = [0 \ \dots \ 0]. \quad (4.8)$$

The condition that x_i is the i -th iterate of the algorithm $\mathcal{M}_\alpha$ can then be expressed as $x_i = Ph_i$ for all $i \in \{0, \dots, N\}$, while the condition $x_* = 0$ is equivalent to $x_* = Ph_*$. Similarly, let $u_i = e_{i+1} \in \mathbb{R}^{N+2}$ for all $i \in \{0, \dots, N\}$, where $\{e_i\}_{i \in \{1, \dots, N+2\}}$ denotes the canonical basis of $\mathbb{R}^{N+2}$, and let $u_* = 0$. Then, we have $g_i = Pu_i$ for all $i \in \{0, \dots, N\}$ and $g_* = Pu_*$. Using these definitions, we obtain:

$$\begin{aligned} \forall i, j \in I, \quad \langle x_i, x_j \rangle &= h_i^\top G h_j, \\ \langle g_i, g_j \rangle &= u_i^\top G u_j, \\ \langle g_i, x_j \rangle &= u_i^\top G h_j. \end{aligned}$$

We further define the vector of function values as $F^\top = [f_0, f_1, \dots, f_N] \in \mathbb{R}^d$, so that $f_i = F^\top u_i$ for all $i \in \{0, \dots, N\}$ and $f_* = F^\top u_*$. Using these representations, the interpolation conditions in Problem (d-PEP-Hölder) can be reformulated for all $i, j \in I$ as:

$$F^\top (u_i - u_j) \geq u_j^\top G (h_i - h_j) + \frac{\beta}{(\beta + 1)L_\beta^{\frac{1}{\beta}}} \left((u_i - u_j)^\top G (u_i - u_j) \right)^{\frac{\beta+1}{2\beta}}.$$

This can be expressed more compactly using the trace operator. For all $i, j \in I$, define the matrices:

$$2A_{i,j}(\alpha) = u_j(h_i - h_j)^\top + (h_i - h_j)u_j^\top, \quad B_{i,j} = (u_i - u_j)(u_i - u_j)^\top.$$

The interpolation conditions can now be written as:

$$F^\top(u_i - u_j) \geq \text{Tr}(A_{i,j}(\alpha)G) + \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} \text{Tr}(B_{i,j}G)^{\frac{\beta+1}{2\beta}}.$$

It remains to show how the term $\text{Tr}(B_{i,j}G)^{\frac{\beta+1}{2\beta}}$ can be formulated in a tractable manner, noting that it is clearly convex as $\text{Tr}(B_{i,j}G)$ is linear in G and $\frac{1+\beta}{2\beta} \geq 1$. To achieve this, we use the three-dimensional power cone, defined for a parameter p as:

$$\mathcal{K}_p := \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 \geq 0, x_2 \geq 0, x_1^p \cdot x_2^{1-p} \geq |x_3| \right\}.$$

When $0 < p < 1$, $\mathcal{K}_p$ is a convex cone. For $0 < \beta < 1$, we have $\frac{\beta+1}{2\beta} > 1$.

Thus, the term $\text{Tr}(B_{i,j}G)^{\frac{\beta+1}{2\beta}}$ can be replaced by a new variable $t_{i,j}$ under the constraint:

$$(t_{i,j}, 1, \text{Tr}(B_{i,j}G)) \in \mathcal{K}_{\frac{2\beta}{\beta+1}}.$$

This is possible as $|\text{Tr}(B_{i,j}G)| = \text{Tr}(B_{i,j}G)$ (matrices $B_{i,j}$ and G are positive semidefinite thus $\text{Tr}(B_{i,j}G) \geq 0$). With this substitution, the interpolation conditions become:

$$\begin{aligned} F^\top(u_i - u_j) &\geq \text{Tr}(A_{i,j}(\alpha)G) + \frac{\beta}{(\beta+1)L_\beta^{\frac{1}{\beta}}} t_{i,j}, \\ (t_{i,j}, 1, \text{Tr}(B_{i,j}G)) &\in \mathcal{K}_{\frac{2\beta}{\beta+1}}, \quad \forall i, j \in I, \end{aligned}$$

which can be handled by a conic interior-point solver such as MOSEK [9]. As shown before, the initial condition $\|x_0 - x_*\|^2 \leq R^2$ can be expressed as:

$$\text{Tr}(A_R G) \leq R^2,$$

where $A_R = u_{N+1}u_{N+1}^\top$. The performance criterion $f_N - f_*$ can be written in the form $b^\top F$ with $b = u_N - u_*$. Using these reformulations, Prob-

lem (d-PEP-Hölder) can be rewritten as follows:

$$\begin{aligned}
& \sup_{G, F} && b^T F + \text{Tr}(CG) \\
& \text{subject to} && \text{Tr}(A_R G) \leq R^2 \\
& && F^\top(u_i - u_j) \geq \text{Tr}(A_{i,j}(\alpha)G) + \frac{\beta}{(\beta+1)L^{\frac{1}{\beta}}} t_{i,j}, \forall i, j \in I. \\
& && (t_{i,j}, 1, \text{Tr}(B_{i,j}G)) \in \mathcal{K}_{\frac{2\beta}{\beta+1}} \forall i, j \in I \\
& && \{x^*, g^*, f^*\} = \{0, 0, 0\} \\
& && G \succeq 0 \\
& && \dim(G) \leq d
\end{aligned}$$

(d-SDP-PEP-holder)

The nonconvex rank constraint $\dim(G) \leq d$ enforces the dimensionality of the original problem. However, in the large-scale setting, this constraint can be safely dropped (see Remark 2.11) without further relaxing Problem (d-SDP-PEP-holder). Dropping this nonconvex rank constraint yields a dimension-independent PEP whose solution provides a valid upper bound on the convergence rate of the fixed-step first-order algorithm under analysis over the set of all L_β -Hölder smooth functions defined by $\mathcal{F}_{0, L_\beta}^\beta = \bigcup_{d \in \mathbb{N}} \mathcal{F}_{0, L_\beta}^\beta(\mathbb{R}^d)$. We present the final form of our convex PEP framework for fixed-step first-order algorithms over Hölder smooth convex functions in the following theorem.

Theorem 4.8. *Consider the class $\mathcal{F}_{0, L_\beta}^\beta(\mathbb{R}^d)$ of L_β -Hölder smooth convex functions with $0 < \beta \leq 1$ and $0 \leq L_\beta < +\infty$, and the fixed-step first-order method $\mathcal{M}_\alpha$. For any $d \in \mathbb{N}$, the optimal solution of the following convex conic program is a valid upper bound on the worst-case performance of algorithm $\mathcal{M}_\alpha$ after N*

steps over the functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$:

$$\begin{aligned}
w_\beta(R, L_\beta, N, \alpha_{i,k}, b) = \sup_{G, F} \quad & b^T F \\
\text{subject to} \quad & \text{Tr}(A_R G) \leq R^2 \\
& F^\top f_i - f_j \geq \text{Tr}(A_{i,j}(\alpha) G) \\
& + \frac{\beta}{(\beta+1)L^{\frac{1}{\beta}}} t_{i,j}, \quad \forall i, j \in I. \\
& (t_{i,j}, 1, \text{Tr}(B_{i,j} G)) \in \mathcal{K}_{\frac{2\beta}{\beta+1}} \\
& \{x^*, g^*, f^*\} = \{0, 0, 0\} \\
& G \succeq 0
\end{aligned} \tag{SDP-PEP-holder}$$

with matrices $A_{ij}(\alpha)$, $B_{i,j}$, A_R and vector b as defined above.

To conclude this section, we present a valid dual formulation of Problem (SDP-PEP-holder). While we do not provide a proof of strong duality, numerical experiments indicate that it holds in practice. This dual formulation can be particularly useful, for instance, in simplifying the algorithm design problem for first-order methods over the functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ (see Chapter 2, Section 2.5).

$$\begin{aligned}
\inf_{\tau, \lambda_{i,j}} \quad & \tau R^2 + \sum_{i,j} s_{i,j} \\
\text{subject to} \quad & \text{Tr}(A_R G) \leq R^2 \\
& \tau A_R + \sum_{i,j} \lambda_{i,j} A_{i,j}(\alpha) + \sum_{i,j} \mu_{i,j} B_{i,j} \succeq 0 \\
& (\delta_{i,j}, s_{i,j}, \mu_{i,j}) \in \mathcal{K}_{\frac{2\beta}{\beta+1}}^*, \quad \forall i, j \in I \\
& \lambda_{i,j} = \frac{\beta}{(\beta+1)L^{\frac{1}{\beta}}} \delta_{i,j}, \quad \forall i, j \in I \tag{dual-SDP-PEP-hölder} \\
& b - \sum_{i,j} \lambda_{i,j} (u_i - u_j) = 0 \\
& \tau \geq 0 \\
& \lambda_{i,j} \geq 0
\end{aligned}$$

where $\mathcal{K}^*_{\frac{2\beta}{\beta+1}}$ is the dual cone of the power cone $\mathcal{K}_{\frac{2\beta}{\beta+1}}$, whose definition is

$$\mathcal{K}^*_p := \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 \geq 0, x_2 \geq 0, \left(\frac{x_1}{p} \right)^p \cdot \left(\frac{x_2}{1-p} \right)^{1-p} \geq |x_3| \right\}.$$

4.4 Worst-case Analysis of Fixed-Step First-Order Algorithms for the Functional class $\mathcal{F}^\beta_{0,L_\beta}(\mathbb{R}^d)$

In this section, we present numerical results on the worst-case convergence of fixed-step first-order algorithms over the class $\mathcal{F}^\beta_{0,L_\beta}(\mathbb{R}^d)$ using the PEP framework developed in the previous section. We compare our findings with existing first-order methods for this class of functions. Our focus is on the memoryless gradient descent algorithm (see Algorithm (MGD)) with step sizes of the form $\frac{R^{1-\beta}\alpha_i}{L_\beta}$. This choice of step sizes is justified by the fact that the worst-case performance of such algorithms exhibits homogeneity properties with respect to L_β and R , which simplifies the analysis. This is stated in the next theorem:

Theorem 4.9 ((L_β, R) -homogeneity). *Define*

$$W_\beta(R, L_\beta, N, \alpha_i) = \max_{f \in \mathcal{F}^\beta_{0,L_\beta}(\mathbb{R}^d)} f(x_N) - f(x_*),$$

as the worst-case performance after N steps of Algorithm (MGD) with step sizes α_i over the functional class $\mathcal{F}^\beta_{0,L_\beta}(\mathbb{R}^d)$, where x_* is an optimal point of f , x_N is the final iterate of Algorithm (MGD)(α_i), and $R = \|x_0 - x_*\|$. Then, the following holds:

$$W_\beta \left(R, L_\beta, N, \frac{R^{1-\beta}\alpha_i}{L_\beta} \right) = L_\beta R^{1+\beta} W_\beta(1, 1, N, \alpha_i).$$

Proof. Given a function f that belongs to $\mathcal{F}^\beta_{0,L_\beta}(\mathbb{R}^d)$, an optimal point x_* of f , an initial iterate x_0 such that $R = \|x_0 - x_*\|$, and the successive iterates $\{x_i\}_{i \in \{1, \dots, N\}}$ generated by (MGD) with step sizes $\frac{R^{1-\beta}\alpha_i}{L_\beta}$ on the function f , define the function $\tilde{f}$ such that:

$$\forall x \in \mathbb{R}^d, \tilde{f}(x) = \frac{1}{L_\beta R^{1+\beta}} f(Rx).$$

Function $\tilde{f}$ is convex since f is convex, and we also have that:

$$\forall x \in \mathbb{R}^d, \nabla \tilde{f}(x) = \frac{1}{L_\beta R^\beta} \nabla f(Rx),$$

which implies that:

$$\begin{aligned} \forall x, y \in \mathbb{R}^d, \|\nabla \tilde{f}(x) - \nabla \tilde{f}(y)\| &= \frac{1}{L_\beta R^\beta} \|\nabla f(Rx) - \nabla f(Ry)\| \\ &\leq \frac{1}{L_\beta R^\beta} L_\beta \|Rx - Ry\|^\beta \quad (f \in \mathcal{F}_{0, L_\beta}^\beta(\mathbb{R}^d)) \\ &\leq \|x - y\|^\beta, \end{aligned}$$

which proves that $\tilde{f}$ belongs to $\mathcal{F}_{0,1}^\beta(\mathbb{R}^d)$. Since x_* is an optimal point of f , then $\tilde{x}_* = \frac{x_*}{R}$ is an optimal point of $\tilde{f}$. Now, define $\tilde{x}_i = \frac{x_i}{R} \forall i \in \{0, \dots, N\}$. We have that $\|\tilde{x}_0 - \tilde{x}_*\| = \frac{1}{R} \|x_0 - x_*\| = 1$, and for all $i \in \{0, \dots, N-1\}$:

$$\begin{aligned} x_{i+1} &= x_i - \frac{R^{1-\beta} \alpha_i}{L_\beta} \nabla f(x_i) \\ &= x_i - \frac{R^{1-\beta} \alpha_i}{L_\beta} \left(L_\beta R^\beta \nabla \tilde{f}(\tilde{x}_i) \right) \quad (\text{definition of } \tilde{f}) \\ &= x_i - \alpha_i R \nabla \tilde{f}(\tilde{x}_i). \end{aligned}$$

Dividing by R , we obtain for all $i \in \{0, N-1\}$:

$$\tilde{x}_{i+1} = \tilde{x}_i - \alpha_i \nabla \tilde{f}(\tilde{x}_i),$$

which proves that $\{\tilde{x}_i\}_{i \in \{0, \dots, N\}}$ are the iterates of (MGD) with step sizes α_i on $\tilde{f}$. Thus, the performance of Algorithm (MGD) with step sizes α_i on $\tilde{f}$ is equal to:

$$\tilde{f}(\tilde{x}_N) - \tilde{f}(\tilde{x}_*) = \frac{1}{L_\beta R^{1+\beta}} (f(x_N) - f(x_*)).$$

For every function $f \in \mathcal{F}_{0, L_\beta}^\beta(\mathbb{R}^d)$, we constructed a function $\tilde{f} \in \mathcal{F}_{0,1}^\beta(\mathbb{R}^d)$ such that the performance of (MGD) on f with step sizes $\frac{R^{1-\beta} \alpha_i}{L_\beta}$ and $\|x_0 - x_*\| = R$ is equal to $L_\beta R^{1+\beta}$ times the performance of (MGD) on $\tilde{f}$ with step sizes α_i and $\|x_0 - x_*\| = 1$. Conversely, for any function $\tilde{f} \in \mathcal{F}_{0,1}^\beta(\mathbb{R}^d)$, we can construct, following a very similar reasoning, a function $f \in \mathcal{F}_{0, L_\beta}^\beta(\mathbb{R}^d)$

defined as:

$$\forall x \in \mathbb{R}^d, f(x) = L_\beta R^{\beta+1} \tilde{f}\left(\frac{x}{R}\right),$$

such that the performance of (MGD) on f with step sizes $\frac{R^{1-\beta}\alpha_i}{L_\beta}$ and $\|x_0 - x_*\| = R$ is equal to $L_\beta R^{1+\beta}$ times the performance of (MGD) on $\tilde{f}$ with step sizes α_i and $\|\tilde{x}_0 - \tilde{x}_*\| = 1$. Thus, we have shown that:

$$W_\beta\left(R, L_\beta, N, \frac{R^{1-\beta}\alpha_i}{L_\beta}\right) = L_\beta R^{1+\beta} W_\beta(1, 1, N, \alpha_i),$$

which concludes the proof. $\square$

Remark 4.10. When analyzing (MGD) with step sizes of the form $\frac{R^{1-\beta}\alpha_i}{L_\beta}$, we can restrict ourselves to solving Problem (SDP-PEP-holder) only for the values $L_\beta = 1$ and $R = 1$, since the upper bound obtained provides a valid upper bound for any values of L_β and R thanks to Theorem 4.9.

We now review key results on the worst-case convergence of first-order methods for L_β -Hölder smooth convex functions. First, we present a lower bound, established by Nemirovsky and Yudin, on the worst-case convergence of first-order methods over the functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$:

Theorem 4.11. [69] *For any sequence of iterates $\{x_i\}_{i \in \{0, \dots, N\}}$ generated by any first-order algorithm on a function f that belongs to $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$, it holds that:*

$$f(x_N) - f(x_*) \geq \mathcal{O}\left(\frac{L_\beta R^{1+\beta}}{N^{\frac{1+3\beta}{2}}}\right),$$

where x_* is an optimal point of f and $\|x_0 - x_*\| \leq R$.

In [73], Nesterov proposed the Universal Gradient Method (UGM), which automatically adapts to the Hölder parameter β and the constant L_β through a line search procedure and achieves the following worst-case convergence bound:

Theorem 4.12. *Assume that function $f : \mathbb{R}^d \mapsto \mathbb{R}$, a differentiable function, belongs to $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ for all $\beta \in [0, 1]$, then after N iterations of the (UGM) method, it holds that:*

$$f(x_N) - f^* \leq \inf_{0 \leq \beta \leq 1} \frac{L_\beta R^{1+\beta}}{N^{\frac{1+\beta}{2}}},$$

where x_* is an optimal point of f and $\|x_0 - x_*\| \leq R$.

Remark 4.13. The previous theorem can be interpreted as follows: for a Hölder smooth convex function f that belongs to several function classes $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ with different parameters β and L_β , UGM automatically adapts to the most favorable description of f . Specifically, it achieves the smallest possible convergence rate among all rates of the form $\frac{L_\beta R^{1+\beta}}{N^{\frac{1+\beta}{2}}}$, corresponding to the classes $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ to which f belongs.

We now leverage the convex PEP framework developed in the previous section to analyze Algorithm (MGD) with step size $\frac{R^{1-\beta}}{L_\beta(N+1)^{\frac{1-\beta}{2}}}$, where N is the total number of iterations. The algorithm is defined by the update rule:

$$x_{i+1} = x_i - \frac{R^{1-\beta}}{L_\beta(N+1)^{\frac{1-\beta}{2}}} \nabla f(x_i). \quad (4.9)$$

Remark 4.14. This choice of step sizes is justified by the fact that $\frac{R^{1-\beta}}{L_\beta(N+1)^{\frac{1-\beta}{2}}}$ interpolates between valid step sizes that ensure the convergence of the method in the extreme cases $\beta = 0$ and $\beta = 1$. Indeed, the intuition behind the power $\frac{1-\beta}{2}$ is that, for the extreme case $\beta = 1$, this power is equal to zero, and we obtain the constant step sizes $\frac{1}{L_1}$, which are widely used step sizes for gradient descent over smooth convex functions. In the other extreme case $\beta = 0$, the power $\frac{1-\beta}{2}$ simplifies to $\frac{1}{2}$, and we obtain the step sizes $\frac{R}{L_0\sqrt{N+1}}$, which are standard for the subgradient method over nonsmooth convex functions.

Remark 4.15. Note that our algorithm is not universal, as its step sizes depend on the Hölder smoothness parameter β . Moreover, our PEP framework takes β as a parameter, which is usually not known in practice. Note that other non-universal algorithms (which are not adaptive with respect to the smoothness parameter β) were derived in [105] and in [19] for the more general setting of Hilbert spaces, both relying on deriving step sizes ensuring a descent property for their algorithms. However, these are not fixed-step algorithms, as the computation of their step sizes depends on quantities that involve the norm of the gradient of the function to optimize, which makes their analysis within a convex PEP framework out of reach and their comparison to our algorithm difficult.

We then compare its performance with the one of the universal gradient method and show that for value of β higher than some threshold, our method has a faster convergence rate than the one of (UGM) but is suboptimal compared to (UGM) for lower values of β .

In Figure 4.1, we provide plots representing the inverse of the upper bound on the worst-case convergence of Algorithm (MGD) with step size $\frac{R^{1-\beta}}{L_\beta(i+1)^{\frac{1-\beta}{2}}}$, obtained using our PEP framework for values of $\beta = 0.3, 0.5, 0.7$. Figures 4.1a, 4.1b, and 4.1c provide strong numerical evidence supporting the fact that our fixed-step first-order algorithm with step sizes $\frac{R^{1-\beta}}{L_\beta(N+1)^{\frac{1-\beta}{2}}}$

converges on functions f that belong to $\mathcal{F}_{0, L_\beta}^\beta(\mathbb{R}^d)$. Recall that, for the numerical experiments, we fix without loss of generality $R = 1$ and $L_\beta = 1$ (see Remark 4.10). We also computed numerical fits for these upper bounds. For $\beta = 0.7$, we have:

$$f(x_N) - f(x_*) \leq L_\beta R^{1+\beta} \frac{1}{3.53N^{0.8756} + 1.163}.$$

For $\beta = 0.5$, we have:

$$f(x_N) - f(x_*) \leq L_\beta R^{1+\beta} \frac{1}{3.129N^{0.7413} + 0.6199}.$$

Finally, for $\beta = 0.3$, we have:

$$f(x_N) - f(x_*) \leq L_\beta R^{1+\beta} \frac{1}{2.621N^{0.5748} + 0.2359}.$$

Remark 4.16. Note that the previous fits are computed by solving the PEP for our algorithm for multiple number of step sizes (up to $N = 50$ in practice) and then fit (using a Matlab toolbox in practice) these numerical data to a model of the form $\frac{1}{\alpha N^\nu + \gamma}$ where α, ν, γ are the parameters to compute.

In Table 4.1, we summarize and compare the convergence rates of our algorithm with those of Nesterov's Universal Gradient Method (UGM). The numerical fits of the upper bounds we obtained and Table 4.1 tend to indicate that the convergence is slower than the rate $O\left(1/K^{\frac{1+\beta}{2}}\right)$ achieved by (UGM) on this class of functions. Indeed, for $\beta = 0.5$ and $\beta = 0.3$, the numerical fits seem to indicate that the convergence is slower. However, the numerical fit for $\beta = 0.7$ tends to support the idea that for sufficiently

smooth functions, our fixed-step first-order algorithm achieves a better convergence rate than (UGM). To confirm this observation, we plot in Figure 4.1 the evolution of the worst-case convergence rate of our algorithm with respect to the level of Hölder-smoothness, which confirms that for values of β higher than some threshold, i.e., $\beta_0 \leq \beta < 1$, our algorithm has a faster convergence rate than that of (UGM). Figure 4.2 shows that the threshold β_0 satisfies $0.5 < \beta_0 < 0.6$. However, Figure 4.1 also highlights that our method is significantly slower for low levels of Hölder-smoothness. For instance, for $\beta = 0.1$, our algorithm exhibits an even slower convergence rate than $\mathcal{O}(1/\sqrt{N})$, which can be achieved using the smaller step size $\frac{1}{\sqrt{N+1}}$. This indicates that our step sizes are too large for low levels of smoothness.

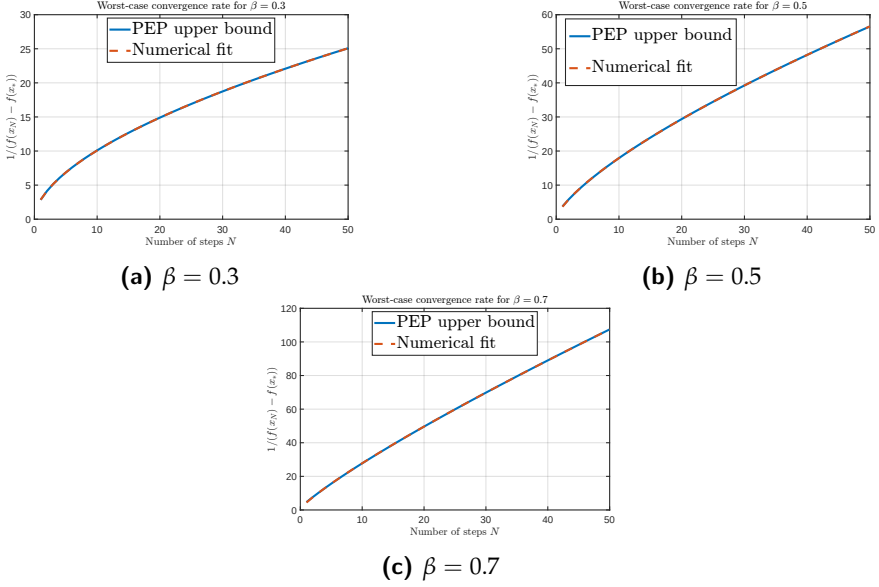


Fig. 4.1 Upper bound on the worst-case convergence of Algorithm (MGD) with step size $\frac{R^{1-\beta}}{L_\beta(N+1)^{\frac{1-\beta}{2}}}$ over the class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ for different values of β .

Hölder smoothness level β	MGD (numerical fits)	UGM [73]
1	$\mathcal{O}(1/N)$	$\mathcal{O}(1/N)$
0.7	$\mathcal{O}(1/N^{0.8756})$	$\mathcal{O}(1/N^{0.85})$
0.5	$\mathcal{O}(1/N^{0.7208})$	$\mathcal{O}(1/N^{0.75})$
0.3	$\mathcal{O}(1/N^{0.5748})$	$\mathcal{O}(1/N^{0.65})$
0	$\mathcal{O}(1/N^{0.5})$	$\mathcal{O}(1/N^{0.5})$

Table 4.1 Comparison of the worst-case convergence rate of Algorithm (MGD) with step size $\frac{R^{1-\beta}}{L_\beta(N+1)^{\frac{1-\beta}{2}}}$ and (UGM) [73] over some function classes $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ with different values of β .

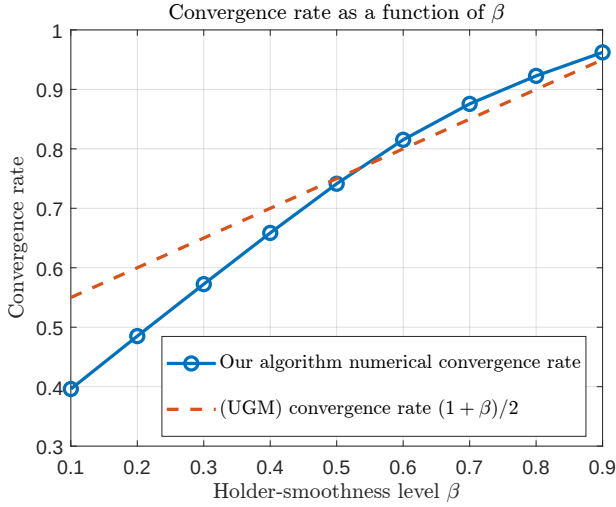


Fig. 4.2 Estimated convergence rates as a function of β for our algorithm, compared to UGM convergence rate $(1 + \beta)/2$.

To conclude this section, we have defined a simple fixed-step first-order algorithm whose convergence on the functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ is supported by strong numerical evidence. However, these same numerical experiments indicate that for a high level of smoothness our algorithm performs slightly better than (UGM) but is significantly worse for low levels of smoothness.

4.5 Conclusion

In this chapter, we extended the PEP framework over Hölder-smooth convex functions, provided necessary $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolation conditions, and used these conditions to derive a convex PEP formulation for the subclass of fixed-step first-order algorithms. We proposed a simple fixed-step first-order algorithm whose step sizes interpolate between those of the classical gradient descent algorithm with constant step size $\frac{1}{L_1}$ for L_1 -smooth convex functions and the subgradient method with step sizes $\frac{R}{L_0\sqrt{N+1}}$ for nonsmooth convex functions with difference of subgradients bounded by L_0 . We provided numerical upper bounds on the worst-case convergence of our algorithm, suggesting a sublinear convergence rate over the functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$. Our algorithm exhibits slightly better performance for high levels of smoothness compared to the universal gradient method but suboptimal convergence for low levels of smoothness.

Open research directions include deriving necessary and sufficient $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$ -interpolation conditions that would provide access to the exact worst-case performance of first-order algorithms over the functional class $\mathcal{F}_{0,L_\beta}^\beta(\mathbb{R}^d)$, designing optimal fixed-step first-order algorithms for this class of problems and extending our convex PEP framework to universal gradient methods.

5

Numerical Design of Optimized First-Order Algorithms

5.1 Introduction

First-order algorithms have become some of the methods of choice for solving modern optimization problems, which are increasingly complex and large-scale, due to their simplicity and low computational cost. However, their effectiveness heavily depends on the choice of step sizes. As a result, finding step sizes that lead to more efficient first-order algorithms has become a major focus of research.

The first results on accelerating the convergence of first-order methods date back to the work of Nemirovsky and Polyak [68, 80], where accelerated methods for quadratic problems were presented. Beyond the quadratic case, efforts to accelerate the convergence of first-order algorithms began with the work of Nemirovsky and Yudin on information-based complexity [69]. Nemirovsky also proposed the first accelerated first-order method for smooth convex optimization, which required a two-dimensional subspace search. Nesterov later resolved this limitation by introducing a computationally ef-

efficient accelerated gradient method for smooth convex functions [70]. Since then, acceleration schemes have been extended to a wide range of problems, including proximal gradient methods [13, 61, 64], random block coordinate descent [72, 36], and stochastic gradient descent [90, 5].

The recent development of the PEP framework, which provides a way to automatically compute upper bounds on the convergence of fixed-step first-order methods in various settings, has simplified the design of optimized first-order methods. This has led to the discovery of new accelerated gradient methods, some of which are information-theoretically optimal for their respective problem classes [55, 95].

Despite these successes, using PEPs to design first-order methods results in nonconvex problem formulations. Consequently, existing approaches have largely relied on heuristic step-size schedules [44, 45, 7, 8], problem-specific relaxations [11, 95, 32, 55, 42, 79] where analytical solutions were found, or computationally intensive branch-and-bound methods (BNB-PEP) [48, 53]. For a more detailed review of recent optimized and optimal first-order algorithms over the classes of smooth (strongly) convex functions, we refer the reader to Section 2.5 of Chapter 2.

In this chapter, our objective is to develop methods for the automatic, computer-assisted numerical design of optimized fixed-step first-order optimization algorithms. We apply these methods to several convex optimization settings and show that our optimized algorithms can achieve faster convergence rates than classical first-order methods in these settings. This chapter is organized as follows:

Organization and contributions. In Section 5.2, we start by defining a generic approach to the design of fixed-step first-order methods. We then present several numerical approaches for automatically designing optimized first-order methods in Section 5.3. In Section 5.4, we compare the efficiency of our methods with the BNB-PEP approach on the benchmark task of designing an optimized fixed-step first-order memoryless gradient descent algorithm over the class of smooth convex functions. Finally, in Section 5.5, we apply our methodologies to numerically design optimized fixed-step first-order algorithms in several different settings, namely for block coordinate-wise algorithms, inexact gradient descent, gradient methods for L_β -Hölder smooth convex functions, and a cyclic gradient descent

algorithm in the setting of linear convergence of the gradient. We provide improved numerical upper bounds on convergence rates for these algorithms compared to existing fixed-step first-order algorithms. Furthermore, our numerical results suggest that recent findings on the accelerated convergence of first-order methods over smooth (strongly) convex functions [7, 8, 45] without using Nesterov momentum-based acceleration schemes can be extended to several other settings.

5.2 Problem Formulation

In this section, we define the problem of designing optimized fixed-step first-order algorithms, as described in (2.8), for certain classes of functions. Consider a class of convex functions $\mathcal{F}(\mathbb{R}^d)$ and an algorithm $\mathcal{M}$. Recall that by defining the set of variables $\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}$ with $I = \{0, \dots, N, *\}$, where x_i for $i \in \{1, \dots, N\}$ represents the iterates of algorithm $\mathcal{M}$, the PEP for $\mathcal{M}$ over the class $\mathcal{F}(\mathbb{R}^d)$ can be written as:

$$\begin{aligned}
& \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} \mathcal{P}(\mathcal{S}_N) \\
& \text{subject to} \quad \mathcal{I}(\mathcal{S}_N) \leq R \\
& \quad \mathcal{S}_N \text{ is } \mathcal{F}(\mathbb{R}^d)\text{-interpolable} \\
& \quad x_0, \dots, x_N \text{ generated by } \mathcal{M}_\alpha \\
& \quad g^* = 0.
\end{aligned} \tag{5.1}$$

where $\mathcal{P}$ is a performance criterion and $\mathcal{I}$ an initial condition that ensures that the worst-case of the algorithm is bounded. Conceptually, the optimal fixed-step first-order algorithm for this class of functions is obtained by solving

$$\begin{aligned}
& \inf_{\alpha} \sup_{\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}} \mathcal{P}(\mathcal{S}_N) \\
& \text{subject to} \quad \mathcal{I}(\mathcal{S}_N) \leq R, \\
& \quad \mathcal{S}_N \text{ is } \mathcal{F}(\mathbb{R}^d)\text{-interpolable}, \\
& \quad x_0, \dots, x_N \text{ generated by } \mathcal{M}_\alpha, \\
& \quad g^* = 0.
\end{aligned} \tag{5.2}$$

We now show that, for a certain type of functional classes, the PEP defined by Problem (5.1) can be reformulated to simplify the algorithm design problem (5.2). The first assumption we make about our class of functions is

that it is invariant under additive shifts in function values and translations in the domain, allowing us to assume, without loss of generality, that $x_* = 0$ and $f_* = 0$. Recall from Section 2.3 that the iterates $\{x_i\}_{i \in \{0, \dots, N\}}$ of the fixed-step first-order algorithm with step sizes α , $\mathcal{M}_\alpha$, defined by equation (2.8), and x_* can be represented using the matrix $P = [g_0, \dots, g_N, x_0]$. Indeed, define the vectors $h_i \in \mathbb{R}^{N+2}$ for $i \in \{0, \dots, N\}$ and h_* as:

$$h_i^\top = [-\alpha_{i,0} \ -\alpha_{i,1} \ \dots \ -\alpha_{i,i-1} \ 0 \ \dots \ 0 \ 1], \quad h_*^\top = [0 \ \dots \ 0],$$

for all $i \in \{0, \dots, N\}$, we have $x_i = Ph_i$ and $x_* = Ph_*$. Similarly, let $u_i = e_{i+1} \in \mathbb{R}^{N+2}$, where $\{e_i\}_{i \in \{1, \dots, N+2\}}$ is the canonical basis of $\mathbb{R}^{N+2}$, and let $u_* = 0$ represent the gradients, such that $g_i = Pu_i$, $\forall i \in \{0, \dots, N\}$, and $g_* = Pu_*$. Defining the matrix $G = P^\top P$, we have:

$$\begin{aligned} \forall i, j \in I, \quad \langle x_i, x_j \rangle &= h_i^\top G h_j, \\ \langle g_i, g_j \rangle &= u_i^\top G u_j, \\ \langle g_i, x_j \rangle &= u_i^\top G h_j. \end{aligned}$$

Then, any linear expression involving only products between x_i, g_i and x_j, g_j for any $i, j \in I$ of the general form:

$$\begin{aligned} &\beta_{i,j}^{(1)} \langle x_i, x_i \rangle + \beta_{i,j}^{(2)} \langle x_i, x_j \rangle + \beta_{i,j}^{(3)} \langle x_j, x_j \rangle + \beta_{i,j}^{(4)} \langle g_i, g_i \rangle + \beta_{i,j}^{(5)} \langle g_i, g_j \rangle + \beta_{i,j}^{(6)} \langle g_j, g_j \rangle \\ &+ \beta_{i,j}^{(7)} \langle g_i, x_i \rangle + \beta_{i,j}^{(8)} \langle g_i, x_j \rangle + \beta_{i,j}^{(9)} \langle g_j, x_i \rangle + \beta_{i,j}^{(10)} \langle g_j, x_j \rangle, \end{aligned}$$

where $(\beta_{i,j}^{(1)}, \dots, \beta_{i,j}^{(10)})$ can be any vector in $\mathbb{R}^{10}$, admits a reformulation:

$$\text{Tr}(A_{i,j}(\alpha)G),$$

with

$$A_{i,j}(\alpha) = \frac{Z_{i,j}(h_i, h_j) + Z_{i,j}(h_i, h_j)^\top}{2} \in \mathbb{S}^{N+2}, \quad (5.3)$$

where

$$\begin{aligned} Z_{i,j}(h_i, h_j) &= \beta_{i,j}^{(1)}(h_i h_i^\top) + \beta_{i,j}^{(2)}(h_i h_j^\top) + \beta_{i,j}^{(3)}(h_j h_j^\top) + \beta_{i,j}^{(4)}(u_i u_i^\top) + \beta_{i,j}^{(5)}(u_i u_j^\top) \\ &+ \beta_{i,j}^{(6)}(u_j u_j^\top) + \beta_{i,j}^{(7)}(u_i h_i^\top) + \beta_{i,j}^{(8)}(u_i h_j^\top) + \beta_{i,j}^{(9)}(u_j h_i^\top) \\ &+ \beta_{i,j}^{(10)}(u_j h_j^\top). \end{aligned}$$

This shows that constraints involving only linear expressions of the products $x_i^\top g_j$, $x_i^\top x_j$, or $g_i^\top g_j$ for any $i, j \in I$ can be written in a convex manner in PEPs as linear matrix inequalities on the Gram matrix G . Consequently, we require that the functional classes under consideration admit necessary interpolation conditions of this form. We formalize this by defining a class of functions that satisfy this assumption, and thus for which the PEPs of fixed-step first-order methods admit a convex formulation. This enables a simpler reformulation of the problem of designing optimized fixed-step first-order methods over these function classes.

Definition 5.1 (SDP-representable classes of functions). Given a class of functions $\mathcal{F}(\mathbb{R}^d)$ that is invariant under additive shifts in function values and translations in the domain, we say that $\mathcal{F}(\mathbb{R}^d)$ is SDP-representable if and only if, for any set $\mathcal{S}_N = \{(x_i, g_i, f_i)\}_{i \in I}$ with $I = \{0, \dots, N, *\}$, where $\{x_i\}_{i \in \{0, \dots, N\}}$ are the iterates generated by the fixed-step first-order algorithm $\mathcal{M}_\alpha$ defined by (2.8), the functional class $\mathcal{F}(\mathbb{R}^d)$ admits necessary interpolation conditions that, when applied to the set $\mathcal{S}_N$, can be written in the form:

$$F^\top (\mu_i u_i - \mu_j u_j) \geq \text{Tr}(A_{i,j}(\alpha)G), \quad \forall i, j \in I, \quad (5.4)$$

where $F^\top = [f_0, f_1, \dots, f_N]$, μ_i for all $i \in I$ are scalars (usually $\mu_i = \mu_j = 1$, for instance the case for the interpolation conditions presented in the previous chapters), h_i, u_i for all $i \in I$ are as defined above, and $A_{i,j}(\alpha)$ is of the form given in equation (5.3).

For any SDP-representable functional class $\mathcal{F}(\mathbb{R}^d)$, PEPs for fixed-step first-order algorithms admit a convex SDP formulation (potentially a relaxation):

Theorem 5.2. *Consider an SDP-representable functional class $\mathcal{F}(\mathbb{R}^d)$, the fixed-step first-order method $\mathcal{M}_\alpha$, a performance criterion of the form $\mathcal{P}(F, G) = b^\top F + \text{Tr}(CG)$, and an initial condition of the form $\mathcal{I}(G)^2 = \text{Tr}(A_R G)$. For any $d \in \mathbb{Z}$, a valid upper bound on the performance after N steps of the fixed-step first-order*

algorithm $\mathcal{M}_\alpha$ over the class of functions $\mathcal{F}(\mathbb{R}^d)$ is given by:

$$\begin{aligned}
w^{\text{SDP}}(R, \alpha, N, b, C, A_R) = & \sup_{G \in \mathbb{S}^{N+2}, f \in \mathbb{R}^{N+1}} b^\top F + \text{Tr}(CG) \\
& \text{such that:} \\
& F^\top (\mu_i u_i - \mu_j u_j) \geq \text{Tr}(A_{i,j}(\alpha)G), \forall i, j \in I, \\
& \text{Tr}(A_R G) - R^2 \leq 0, \\
& G \succeq 0,
\end{aligned} \tag{SDP-PEP-Generic}$$

where μ_i, h_i, u_i for all $i \in I$, and $A_{ij}(\alpha)$ are of the form (5.3), as defined in Definition 5.1.

Proof. The result follows from the same reasoning explained, for instance, in Chapter 2, Section 2.3, for the class of L -smooth μ -strongly convex functions $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$, which satisfies the assumptions of Definition 5.1. The main idea is to replace the $\mathcal{F}(\mathbb{R}^d)$ -interpolability constraint in Problem (5.1) with the interpolation conditions defined in (5.4). $\square$

Using the generic convex PEP framework given in Theorem 5.2, we can now formulate the problem of designing optimized fixed-step first-order algorithms for any SDP-representable functional class as the following min-max problem:

$$\begin{aligned}
& \inf_{\alpha_{i,k}} \sup_{F, G} b^\top F + \text{Tr}(CG) \\
& \text{subject to } \text{Tr}(A_R G) \leq R^2 \\
& F^\top (\mu_i u_i - \mu_j u_j) \geq \text{Tr}(A_{i,j}(\alpha)G), \forall i, j \in I \\
& G \succeq 0.
\end{aligned}$$

Remark 5.3. Note that we do not require the interpolation conditions (5.4) in Definition 5.1 to be sufficient. This implies that the convex PEP formulation in Theorem (5.2) may be a relaxation, which can, in turn, induce a relaxation in the algorithm design problem formulation derived next. As a result, solving it does not guarantee the optimality of the algorithm for the considered functional class.

Note that in this min-max problem, the inner optimization problem corresponds to the SDP Problem (SDP-PEP-Generic), which computes a valid upper bound on the performance of the fixed-step first-order algorithm $\mathcal{M}_\alpha$. Hence, solving this min-max problem corresponds to finding the fixed-step

first-order algorithm with the lowest upper bound on its performance over the class $\mathcal{F}(\mathbb{R}^d)$. Since min-max optimization problems are typically hard to solve, we can further simplify the formulation by transforming it into a single minimization problem. This can be achieved by computing the Lagrangian dual of Problem (SDP-PEP-Generic) (i.e., dualizing the inner problem). The dual of Problem (SDP-PEP-Generic) takes the form:

$$\begin{aligned}
w^{\text{dual}}(\alpha) = \inf_{\tau, \lambda_{i,j}} \quad & \tau R^2 \\
\text{subject to} \quad & \tau A_R - C + \sum_{i,j \in I} \lambda_{i,j} A_{i,j}(\alpha) \succeq 0 \\
& b - \sum_{i,j} \lambda_{i,j} (\mu_j u_j - \mu_i u_i) = 0 \quad (\text{dual-PEP-generic}) \\
& \lambda_{i,j} \geq 0, \forall i, j \in I \\
& \tau \geq 0
\end{aligned}$$

Remark 5.4. Note that if strong duality does not hold, then using the Lagrangian dual of Problem (SDP-PEP-Generic) still yields a valid upper bound on the worst-case performance of the algorithm. However, this may introduce a relaxation in the formulation of the corresponding algorithm design problem. Experimentally, strong duality holds for every explicit PEP formulation considered in the previous chapters of this thesis.

Thus, the problem of designing fixed-step first-order algorithms over an SDP-representable class of functions $\mathcal{F}(\mathbb{R}^d)$ takes the simpler form:

$$\begin{aligned}
\inf_{\alpha_{i,k}, \tau, \lambda_{i,j}} \quad & \tau R^2 \\
\text{subject to} \quad & \mathcal{A}(\tau, \lambda, \alpha) \succeq 0 \\
& b - \sum_{i,j} \lambda_{i,j} (\mu_j u_j - \mu_i u_i) = 0 \quad (\text{Algo-design-generic}) \\
& \lambda_{i,j} \geq 0, \forall i, j \in I \\
& \tau \geq 0
\end{aligned}$$

where $\mathcal{A}(\tau, \lambda, \alpha) = \tau A_R - C + \sum_{i,j \in I} \lambda_{i,j} A_{i,j}(\alpha)$. We obtain a nonconvex conic formulation of the algorithm design problem for any SDP-representable class of functions, similar to the formulations derived in [55, 97, 48] for the

class of convex smooth functions.

Remark 5.5. The matrix $\mathcal{A}(\tau, \lambda, \alpha)$ is linear in the variables λ when the α are fixed. However, it is nonlinear in both α and λ together and potentially nonlinear in the variables α when the λ are fixed. This implies that the constraint $\mathcal{A}(\tau, \lambda, \alpha) \succeq 0$ is a nonlinear matrix inequality. In general, optimization problems involving nonlinear matrix inequalities are NP-hard to solve [101]. Matrix $\mathcal{A}(\tau, \lambda, \alpha)$ is always linear in τ .

5.3 Numerical Design Methods for First-Order Optimization Algorithms

We present several methods for solving the problem of designing optimized first-order methods, namely: an alternating minimization approach, a first-order method, and a sequential linearization method.

Alternating Minimization (AM). Although Problem (Algo-design-generic) is a nonlinear SDP, in many cases of interest, it exhibits a bilinear structure. Specifically, in these cases, the matrix $\mathcal{A}(\tau, \lambda, \alpha)$ is also linear with respect to the variables α when the λ are fixed. This is notably the case when designing fixed-step first-order algorithms for the class of L -smooth μ -strongly convex functions $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ with $0 \leq \mu < L < +\infty$, as shown in [55, 56, 95]. In these instances, when either the variables α or λ are fixed, Problem (Algo-design-generic) reduces to a linear SDP. Therefore, it is possible to alternately optimize with respect to λ and α by solving linear SDPs, following the approach in [18]. Let us now define the subproblem when the λ are fixed:

$$\begin{aligned} & \inf_{\alpha_{i,k}, \tau} && \tau R^2 \\ & \text{subject to} && \mathcal{A}(\tau, \lambda, \alpha) \succeq 0 \\ & && b - \sum_{i,j} \lambda_{i,j} (\mu_j u_j - \mu_i u_i) = 0 \\ & && \tau \geq 0 \end{aligned} \quad (\text{Opt-steps})$$

Note that variable τ is optimized at both steps of our algorithm. We can now define our alternating minimization algorithm for the design of first-order methods:

Algorithm 11 Alternating minimization for optimizing first-order algorithms (AM)

```

1: Input: Initial step-sizes  $\alpha_{i,k}^{(0)}$ , Number of iterations  $T$ 
2: Output:  $\alpha_{i,k}^{(N)}$ 
3: for  $t = 1$  to  $T$  do
4:   Compute  $\lambda_{i,j}^{(t)}$  by solving Problem (dual-PEP-generic) for  $\mathcal{M}_{\alpha^{(t-1)}}$ 
5:   Compute  $\alpha_{i,k}^{(t)}$  by solving problem (Opt-steps) with  $\lambda_{i,j} = \lambda_{i,j}^{(t)}$ 
6: end for

```

Remark 5.6. Note that when strong duality holds (which has experimentally been the case for every explicit PEP formulation considered in this thesis), the variables $\lambda_{i,j}$ in Problem (dual-PEP-generic) characterize the proof of the upper bound on the performance of the algorithm $\mathcal{M}_\alpha$. Informally, we obtain the upper bound by multiplying each constraint in Problem (SDP-PEP-Generic) by its corresponding dual variable $\lambda_{i,j}$ and summing all the resulting inequalities. For a more detailed explanation of PEP-assisted convergence proofs, we refer the reader to [39]. Thus, our alternating minimization approach can be understood as alternatively finding the optimal upper bound on the performance of a given algorithm along with a valid proof with respect to the set of constraints used in Problem (SDP-PEP-Generic), and then finding the best algorithm such that the proof of the previous upper bound remains valid for the new algorithm.

Remark 5.7. Note that although our alternating minimization approach does yield better step-sizes compared to the initial step-sizes, we have no guarantee that it converges to a global minimum nor even to a local one. Indeed consider the simple example of optimizing two steps of (MGD) over L -smooth convex functions. In the next figure, we provide a contour plot for step sizes between 0 and $\frac{2}{L}$ of the worst-case performance obtained by PEP and highlight the critical points of the worst-case over the space of step sizes (one global minimum and one local minimum), we also place on this figure the successive iterates of alternating minimization initialized with the classical step sizes $\frac{1}{L}$. Figure 5.1 shows that alternating minimization gets stuck at a point of non-smoothness at the boundary between two regions and does not converge to either minimum.

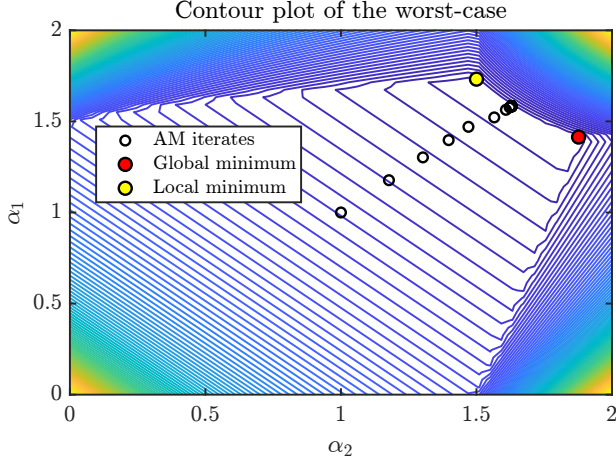


Fig. 5.1 AM iterates are not guaranteed to converge to a minimum

First-order method (FOM). Our second approach is to apply a first-order method to the function $w^{\text{dual}}(\alpha)$ directly. This requires computing the gradients, when well-defined, of w^{dual} with respect to the step sizes $\alpha_{i,k}$. Since the step sizes $\alpha_{i,k}$ influence $w^{\text{dual}}(\alpha)$ only through the vectors h_i in the matrices $A_{i,j}(\alpha)$ (see Problem dual-PEP-generic), computing the gradients of w^{dual} with respect to the step sizes requires characterizing the variations of w^{dual} under small perturbations of the matrix

$$\tau A_R - C + \sum_{i,j \in I} \lambda_{i,j} A_{i,j}(\alpha).$$

To this end, we consider the perturbed problem:

$$\begin{aligned} \phi(B, \alpha) = \inf_{\tau, \lambda_{i,j}} \quad & \tau R^2 \\ \text{subject to} \quad & \tau A_R - C + \sum_{i,j \in I} \lambda_{i,j} A_{i,j}(\alpha) + B \succeq 0 \\ & b - \sum_{i,j} \lambda_{i,j} (\mu_j u_j - \mu_i u_i) = 0 \\ & \lambda_{i,j} \geq 0 \\ & \tau \geq 0 \end{aligned}$$

(Perturbed-dual-PEP)

Theorem 5.8. [15, Proposition 5.80] The function $\phi(., \alpha)$ is convex over the set of positive semidefinite matrices and:

$$\partial_B \phi(0, \{\alpha\}) = \{-G : G \text{ is a solution of Problem (SDP-PEP-Generic)}\}$$

This Theorem relies on the fact that the dual of Problem (dual-PEP-generic) is Problem (SDP-PEP-Generic).

Remark 5.9. Note that, as a consequence of the previous theorem, if Problem (SDP-PEP-Generic) admits a unique solution, the function $\phi(., h_i)$ is differentiable at zero.

Applying the chain rule, we have that:

Theorem 5.10. if w^{dual} is differentiable then the gradient is equal to the vector:

$$\nabla w^{dual}(\alpha) = \left(\frac{\partial w^{dual}}{\partial \alpha_{1,0}}(\alpha), \dots, \frac{\partial w^{dual}}{\partial \alpha_{N,N-1}}(\alpha) \right) \quad (5.5)$$

$$\frac{\partial w^{sdp}}{\partial \alpha_{m,n}}(\alpha) = \text{Tr} \left(-G \frac{\partial \mathcal{A}}{\partial \alpha_{m,n}} \right),$$

where G is the unique (see remark 5.9) solution of Problem (dual-PEP-generic), $\mathcal{A}$ is defined by:

$$\mathcal{A} = \tau A_R - C + \sum_{i,j \in I} \lambda_{i,j} A_{i,j}(\alpha)$$

and $\frac{\partial \mathcal{A}}{\partial \alpha_{m,n}}$ is the matrix such that:

$$\left(\frac{\partial \mathcal{A}}{\partial \alpha_{m,n}} \right)_{i,j} = \frac{\partial \mathcal{A}_{i,j}}{\partial \alpha_{m,n}}$$

Remark 5.11. In practice, we use the vector computed in equation (5.5), even though we cannot guarantee that it is a gradient everywhere or even an ascent direction. Numerical observations show that usually all possible step sizes choices for an algorithm can be divided into finitely many regions within which the PEP admits a unique solution, making w^{dual} differentiable (see remark 5.9) on these regions. Multiple solutions for the PEP may occur only on the boundaries between these regions on which the function w^{dual} can become non-differentiable (see the master theses [24, 34]). Furthermore, our first-order method yields satisfactory results on the benchmark application presented in Section 5.4.

We can now present our first-order approach for the design of optimized first-order algorithms:

Algorithm 12 First-order method (FOM) for the design first-order algorithms

- 1: **Input:** Initial step-sizes $\alpha_{i,k}^{(0)}$, Number of iterations T , a list of decreasing steps $\{\beta_t\}_{[1,\dots,T]}$
 - 2: **Output:** $\alpha_{i,k}^{(T)}$
 - 3: **for** $t = 1$ to T **do**
 - 4: Compute $\frac{\partial w^{\text{sdp}}}{\partial \alpha_{m,n}}(\alpha^{(t-1)})$ using Theorem 5.10
 - 5: Update the step-sizes: $\alpha_{m,n}^{(t)} = \alpha_{m,n}^{(t-1)} - \beta_t \frac{\partial w^{\text{sdp}}}{\partial \alpha_{m,n}}(\{\alpha_{i,k}^{(t-1)}\})$
 - 6: **end for**
-

Sequential Linearization Method (SLM). We now present a third method to optimize the step sizes of first-order algorithms, closely linked to sequential semidefinite programming [23, 35]. The central idea is to iteratively linearize Problem (Algo-design-generic) around a current set of decision variables $\{\tau, \lambda, \alpha\}$. To simplify notations, we define the vector of decision variables as $x = \{\tau, \lambda_{i,j}, \alpha_{i,k}\}$ and denote by x_i the i -th component of vector x . Using this notation, Problem (Algo-design-generic) takes the form:

$$\begin{aligned}
 \inf_x \quad & c^\top x \\
 \text{subject to} \quad & \mathcal{A}(x) \succeq 0 \\
 & b - d^\top x = 0 \\
 & a_i^\top x \geq 0, \forall i \in \{1, \dots, N^2 + 1\}
 \end{aligned} \tag{5.6}$$

Note that all the constraints of the problem are linear except for $\mathcal{A}(x)$. Our method starts by choosing a first feasible point $x^{(0)}$ and iteratively generate

a sequence of iterates $x^{(t)}$, by solving the tangent problem:

$$\begin{aligned}
& \min_{\Delta x} \quad c^\top \Delta x \\
& \text{subject to} \quad \mathcal{A}(x^{(t)}) + \sum \frac{\partial \mathcal{A}}{\partial x_i}(x^{(t)}) \Delta x_i \succeq 0 \\
& \quad b - d^\top (x^{(t)} + \Delta x) = 0 \\
& \quad a_i^\top (x^{(t)} + \Delta x) \geq 0
\end{aligned} \tag{OptAlgo-linear}$$

where $[\frac{\partial \mathcal{A}}{\partial x_i}(x^{(t)})]_{m,n} = \frac{\partial [\mathcal{A}(x^{(t)})]_{m,n}}{\partial x_i}$.

Remark 5.12. Our approach is very similar to sequential semidefinite programming (SQP); the only difference is that, typically, the tangent problem considered in SQP methods includes a quadratic regularization term $\frac{1}{2} \Delta x^\top \tilde{H}_t \Delta x$ in the objective, where the matrix $\tilde{H}_t$ is a positive semidefinite approximation of the Hessian matrix of the Lagrangian of the problem. In our case, the Hessian of the Lagrangian of Problem (5.6) is given by

$$H_k = \nabla_x^2 \text{Tr}(-G\mathcal{A}(x^{(t)}))$$

where G is the semidefinite matrix solution of Problem (SDP-PEP-Generic). We choose not to include such regularization since it would negatively impact the computational efficiency of our method. The number of variables in x increases with the number of step sizes to optimize, which, in turn, increases the dimension of the Hessian matrix to consider.

Ideally, we would like to perform the update $x^{(t+1)} = x^{(t)} + \Delta x^{(t)}$, where $\Delta x^{(t)}$ is the optimal solution of Problem (OptAlgo-linear). However, in practice, this step is not always valid, as it does not necessarily ensure a decrease in the objective function. To address this issue, we incorporate a trust-region method into our algorithm. The trust-region approach adaptively bounds the search space of subproblem (OptAlgo-linear) within a safe region—defined by a radius D —where our linear approximation of the problem remains accurate and the solution of subproblem (OptAlgo-linear) provides an update that decreases the objective function. This is achieved by adding a constraint on the norm of the variable Δx :

$$\begin{aligned}
& \underset{\Delta x}{\text{minimize}} && c^\top \Delta x \\
& \text{subject to} && \mathcal{A}(x^{(t)}) + \sum \frac{\partial \mathcal{A}}{\partial x_i}(x^{(t)}) \Delta x_i \geq 0 \\
& && b - d^\top (x^{(t)} + \Delta x) = 0 \\
& && a_i^\top (x^{(t)} + \Delta x) \geq 0 \\
& , && \frac{1}{2} \Delta x^\top \Delta x \leq D
\end{aligned} \tag{SLM-Subproblem}$$

The next step in the trust-region procedure is to define an update rule for the radius and the current iterate, based on the performance of the update computed by solving Problem (SLM-Subproblem). Let $D^{(t)}$ and the current iterate $x^{(t)} = [\tau^{(t)}, \lambda^{(t)}, \alpha^{(t)}]$ be given, along with the update $\Delta x^{(t)} = [\Delta \tau^{(t)}, \Delta \lambda^{(t)}, \Delta \alpha^{(t)}]$. To assess the update's performance, we introduce the performance criterion P , defined as the ratio of the actual improvement—represented by the difference in upper bounds on the performances of the first-order algorithms $\mathcal{M}(\alpha_{i,k}^{(t)})$ and $\mathcal{M}(\alpha^{(t)} + \Delta \alpha^{(t)})$ —given by

$$\Delta W = w^{\text{dual}}(\alpha^{(t)} + \Delta \alpha^{(t)}) - w^{\text{dual}}(\alpha^{(t)}),$$

to the expected improvement given by $\Delta \tau^{(t)} R^2$:

$$P = \frac{\Delta W}{\Delta \tau^{(t)} R^2}.$$

If $P \geq 0.9$, the update is considered good, and we set $D^{(t+1)} = 2D^{(t)}$, $\alpha_{i,k}^{(t+1)} = \alpha_{i,k}^{(t)} + \Delta \alpha_{i,k}^{(t)}$. If $P \leq 0.1$, the update's performance is poor, so we set $D^{(t+1)} = D^{(t)}/2$ and $\alpha_{i,k}^{(t+1)} = \alpha_{i,k}^{(t)}$. In all other cases where P is between 0.1 and 0.9, we set $\alpha_{i,k}^{(t+1)} = \alpha_{i,k}^{(t)} + \Delta \alpha_{i,k}^{(t)}$ while keeping $D^{(t+1)} = D^{(t)}$. This process is formalized in the following algorithm:

Algorithm 13 Trust Region Procedure

- 1: **Input:** trust-region radius $D^{(t)}$, current iterate $x^{(t)}$, solution $\Delta x^{(t)} = [\Delta \tau^{(t)}, \Delta \lambda_{i,j}^{(t)}, \Delta \alpha_{i,k}^{(t)}]$ of Problem (SLM-Subproblem)
 - 2: **Output:** $D^{(t+1)}$, $\alpha_{i,k}^{(t+1)}$
 - 3: Compute $\Delta W = w^{\text{dual}}(\alpha^{(t)} + \Delta \alpha^{(t)}) - w^{\text{dual}}(\alpha^{(t)})$
 - 4: Compute the performance criterion $P = \frac{\Delta W}{\Delta \tau^{(t)} R^2}$
 - 5: **if** $P \geq 0.9$ **then**
 - 6: $D^{(t+1)} = 2D^{(t)}$ and $\alpha_{i,k}^{(t+1)} = \alpha_{i,k}^{(t)} + \Delta \alpha_{i,k}^{(t)}$
 - 7: **else if** $P \leq 0.1$ **then**
 - 8: $D^{(t+1)} = \frac{D^{(t)}}{2}$ and $\alpha_{i,k}^{(t+1)} = \alpha_{i,k}^{(t)}$
 - 9: **else**
 - 10: $\alpha_{i,k}^{(t+1)} = \alpha_{i,k}^{(t)} + \Delta \alpha_{i,k}^{(t)}$
 - 11: **end if**
-

We now present our full algorithm as follows:

Algorithm 14 SLM for optimizing first-order algorithms

- 1: **Input:** Initial step-sizes $\alpha_{i,k}^{(0)}$, initial trust-region radius $D^{(0)}$, Number of iterations T
 - 2: **Output:** $\alpha_{i,k}^{(T)}$
 - 3: **for** $t = 1$ to T **do**
 - 4: Compute $\mathcal{A}(x^{(t)})$, $\lambda_{i,j}^{(t)}$, $\tau^{(t)}$ by solving problem (dual-PEP-generic)
 - 5: Form the vector $x^{(t)} = [\tau^{(t)}, \lambda^{(t)}, \alpha^{(t)}]$
 - 6: Compute $\Delta x^{(t)} = [\Delta \tau^{(t)}, \Delta \lambda^{(t)}, \Delta \alpha^{(t)}]$ by solving (SLM-Subproblem)
 - 7: Compute $D^{(t+1)}$ and $\alpha_{i,k}^{(t+1)}$ using Algorithm (13).
 - 8: **end for**
-

5.4 Comparison on the benchmark of Optimizing Memoryless Gradient Descent

In this section, we compare the efficiency of our three methods with the BNB-PEP approach developed in [48] for the task of optimizing the step sizes of memoryless gradient descent (MGD) on smooth convex functions. Thanks to the homogeneity result of Theorem 2.15, we can set the smoothness constant to $L = 1$ and the initial distance to a minimizer to $R = 1$

without loss of generality. In the next figure, we provide upper bounds on the worst-case convergence rates for the memoryless gradient descent with the step sizes given in [48] and with the step sizes obtained using each of our three methods: Alternating Minimization (AM), the First-Order Method (FOM), and our Successive Linearization Method (SLM). Note that the algorithm design approaches discussed here all depend on the total number of iterations N of memoryless gradient descent. The plot shown in the next figure is obtained by running each method for each value of N , yielding optimized step sizes specific to that number of iterations. For each case, we compute an upper bound on the performance using PEP, which is then plotted with respect to N .

We run our methods for a maximum of $T = 1000$ iterations and use a stopping condition where we exit the optimization process when the improvement in the upper bound on the worst-case is less than 10^{-7} , and the norm of the difference between two successive updates of the step sizes is less than 10^{-4} . We initialize our step sizes with the classical constant step-size strategy, setting $\alpha_i^{(0)} = \frac{1}{L}$.

Remark 5.13. The BNB-PEP approach consists of two steps. The first step involves solving the algorithm design problem locally by framing it as a nonconvex QCQP problem (similar to our PEP formulations but with substituting the positive semidefinite matrix G using a Cholesky decomposition $G = ZZ^T$ and imposing suitable conditions on the new variable Z) and then using the local solution as a warm-start for the Branch-and-Bound procedure. The full BNB-PEP approach including the Branch-and-Bound procedure should converge to global optimum whereas our methods are only local optimization methods. Note that the BNB-PEP approach can also be used as a local optimization heuristic by forgoing the branch and bound procedure (which was done by the authors in [48] on this benchmark task for a number of iteration higher than 25).

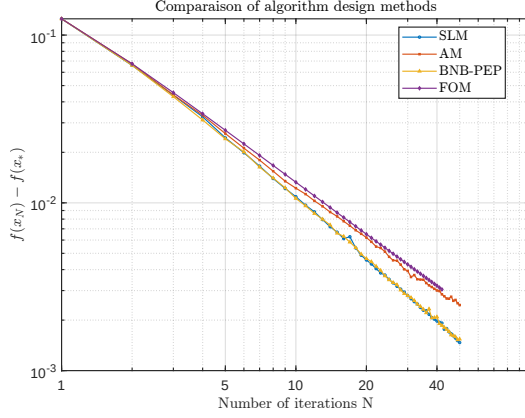


Fig. 5.2 Worst-case convergence of memoryless gradient descent (MGD) with the optimized step-sizes given the BNB-PEP method [48], our Alternating Minimization (AM), First-Order method (FOM) and Successive Linearization method (SLM)

Figure 5.2 shows that the AM and FOM methods are suboptimal compared to BNB-PEP. As discussed in the previous section, AM can get stuck in nonsmooth regions and fail to converge to an optimum. FOM can also suffer from this issue, as it relies on computing vectors that are gradients of the objective where it is differentiable. Consequently, in nonsmooth regions, the updates do not guarantee a decrease in the objective. Experimentally, we observed that the convergence of these two methods depends significantly on the initialization. For some initializations, the methods may fail to converge to an optimum, while for better initializations (i.e a good first guess of optimized step sizes) they converge to an optimal point. Moreover, using different stopping criteria that allow these methods to run for more iterations can improve the results. SLM, on the other hand, is competitive with the BNB-PEP approach, even outperforming it in many instances, as shown in Table 5.1. Note that the step sizes provided by the BNB-PEP method are claimed to be optimal up to $N = 25$. However, Table 5.1 shows that this is not the case, suggesting that the numerical experiments reported in [48] using the BNB-PEP approach may suffer from numerical issues or instabilities. In Table 5.2, we present the optimized step-sizes we obtain using SLM for the first few iterations up to $N = 8$.

Number of iterations N	SLM	BNB-PEP
6	0.019895	0.020098
8	0.013962	0.014055
9	0.012184	0.012282
13	0.007946	0.007984
14	0.007204	0.007376
16	0.006120	0.006311
18	0.005382	0.005404
19	0.004869	0.004941
20	0.004563	0.004652
21	0.004318	0.004456
22	0.004055	0.004185
23	0.003818	0.003954
26	0.003333	0.003347
27	0.003181	0.003247
31	0.002682	0.002722
32	0.002576	0.002618
34	0.002379	0.002411
35	0.002290	0.002318
37	0.002165	0.002330
39	0.002027	0.002069
40	0.001974	0.002089
43	0.001763	0.001808
45	0.001703	0.001709
47	0.001611	0.001616
48	0.001562	0.001589
49	0.001508	0.001526
50	0.001470	0.001541

Table 5.1 Instances where our Successive Linearization Method (SLM) outperforms BNB-PEP [48]

Number of Iterations N	Optimized step-sizes
1	[1.500]
2	[1.414, 1.877]
3	[1.414, 1.601, 2.189]
4	[1.414, 1.601, 1.702, 2.459]
5	[1.414, 1.601, 1.702, 3.526, 1.500]
6	[1.414, 1.601, 1.702, 4.116, 1.732, 1.500]
7	[1.414, 1.601, 1.702, 4.552, 1.414, 2.414, 1.500]
8	[1.414, 1.601, 2.261, 1.414, 5.387, 1.414, 2.414, 1.500]

Table 5.2 Optimized step-sizes for memoryless gradient descent obtained using SLM

Remark 5.14. Note that the step sizes in Table 5.2 seem to benefit from a similar structure to one of the step sizes proposed in [8, 45] but do not match these schedules. However this similar structure might indicate that these step sizes can be retrieved within the framework proposed in [45] of composing smaller step sizes sequences to inductively construct longer ones.

Given the results of Figure 5.2 and Table 5.1, we now focus solely on our SLM approach. Next, we compare the computational efficiency of our SLM approach with that of BNB-PEP. In Table 5.3, we compare the solver time of the first step of the BNB-PEP procedure (i.e, without the branch and bound procedure, using only the local optimization step on the QCQP) with our SLM approach. We choose IPOPT as the QCQP solver and MOSEK for our SLM approach. We run both methods for a maximum of $T = 1000$ iterations.

Number of iterations N	SLM (s)	BNB-PEP (IPOPT) (s)
5	0.13	10.05
10	0.78	11.87
15	2.06	13.05
20	7.63	61.61
25	25.23	131.59 (2.19 min)
30	30.43	250.16 (4.17 min)
35	48.48	963.11 (16.05 min)
40	147.34 (2.46 min)	3158.25 (52.64 min)
45	174.54 (2.91 min)	717.37 (11.96 min)
50	371.76 (6.20 min)	1294.14 (21.57 min)

Table 5.3 Comparison of solver run times for BNB-PEP using IPOPT for local optimization and our SLM approach.

Table 5.3 indicates that our SLM approach appears to be computationally more efficient than the local optimization (using IPOPT) performed in the BNB-PEP method, especially for larger numbers of step sizes to optimize. Note that both methods in most cases converge before reaching the maximum number of iterations $T = 1000$. In [48], the authors provide runtimes for the full BNB-PEP method (including the Branch-and-Bound procedure) on this task. Starting from $N = 10$ iterations, the full BNB-PEP method already takes more than one day to run on a supercomputer. To further demonstrate the computational efficiency of our method, we run it respectively for a maximum of optimization steps equal to $T = 10$ and $T = 100$. We set the maximum iteration parameter of the IPOPT solver to the same values. In the next figure, we provide the worst-case performance rates using the step sizes obtained under these limitations.

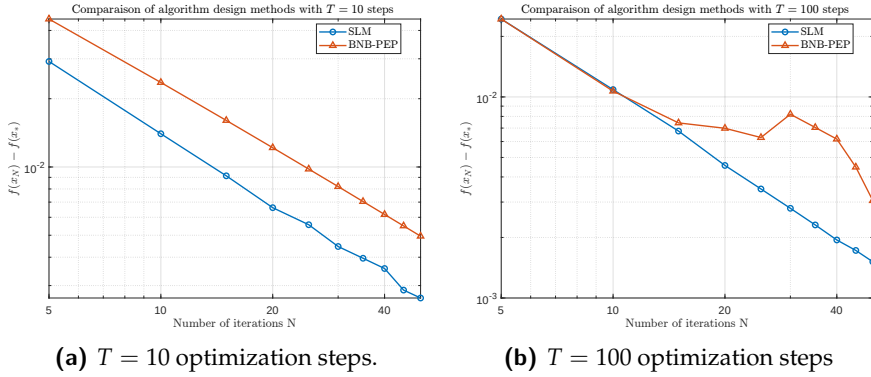


Fig. 5.3 Worst-case convergence of memoryless gradient descent (MGD) with the optimized step sizes obtained using the BNB-PEP method with the local solver IPOPT, and with the step sizes from our Successive Linearization Method (SLM) under the limitation of 100 optimization steps.

Figure 5.3 show that, under the constraint of performing few steps to optimize the algorithm ($T = 10$ and $T = 100$ steps), our SLM approach outperforms the BNB-PEP method (with IPOPT). In the next tables, we provide the solver runtimes for performing $T = 10$ and $T = 100$ steps for each method.

Number of iterations N	SLM (s)	BNB-PEP (IPOPT) (s)
5	0.11	0.024
10	0.17	0.10
15	0.54	0.31
20	1.26	0.86
25	2.43	1.94
30	5.43	4.32
35	9.14	8.90
40	11.47	16.73
45	29.60	26.93
50	45.76	44.68

Table 5.4 Comparison of solver runtimes for performing $T = 10$ steps using SLM and BNB-PEP (IPOPT).

Number of iterations N	SLM (s)	BNB-PEP (IPOPT) (s)
5	0.17	0.10
10	0.95	0.64
15	2.66	1.14
20	11.11	3.07
25	15.91	7.95
30	37.81	15.60
35	54.63	31.22
40	83.68 (1.39 min)	59.12
45	136.97 (2.28 min)	84.67 (1.41 min)
50	193.35 (3.22 min)	138.66 (2.31 min)

Table 5.5 Comparison of solver runtimes for performing $T = 100$ steps using SLM and BNB-PEP (IPOPT).

Tables 5.4 and 5.5 show that for fewer numbers of iterations ($T = 10$ and $T = 100$) our method has a slightly slower solver run time compared to BNB-PEP. Combined with the numerical results in Figure 5.3 and the solver run times until convergence in Table 5.3, this suggests that our SLM approach has a slower run time per iteration but requires significantly fewer steps to converge.

To summarize this section, we presented numerical experiments supporting the competitiveness of our Successive Linearization Method (SLM) with the BNB-PEP approach from [48] on optimizing step sizes for memoryless gradient descent over smooth convex functions. Moreover, our results suggest that SLM requires fewer steps to converge and performs better under restrictions on the number of steps, indicating overall higher efficiency. In the sequel, we will use SLM to design optimized first-order algorithms in various settings.

5.5 Numerical Design of First-Order Methods using SLM

In this section, we apply the SLM method presented above to solve the algorithm design problem and develop fixed-step first-order methods for various convex optimization settings. Numerical results suggest that our approach yields optimized methods that often achieve accelerated convergence rates. For all our experiments, we run our successive linearization method for a maximum of $T = 1000$ steps and terminate the optimization process if the improvement in the upper bound on the worst-case performance falls below 10^{-7} and the norm of the difference between successive

step-size updates is less than 10^{-4} .

Remark 5.15. Note that in this section, the numerical convergence rates presented for the optimized algorithms we design are obtained by solving the Performance Estimation Problem for the algorithm over a range of numbers of iterations N , and then fitting a model of the form $\frac{1}{\alpha N^\nu + \beta}$ to these data.

Design of optimized block coordinate-wise algorithms. We consider the setting of Chapter 3 and apply our algorithm design methodology to tune the step sizes of block coordinate-wise algorithms (see Definition 3.10). The generic framework previously introduced for algorithm optimization extends naturally to this setting. The main difference is that the PEP SDP formulations involve multiple semidefinite matrix variables, which does not hinder the application of the methods presented in this chapter. We focus on cyclic algorithms and begin with the subclass of memoryless cyclic coordinate descent algorithms.

Recall from Chapter 3 that to define block coordinate descent algorithms, we consider a partition of the space $\mathbb{R}^d$ into p subspaces:

$$\mathbb{R}^d = \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_p},$$

with the corresponding selection matrices $U_\ell \in \mathbb{R}^{d \times d_\ell}$, such that:

$$(U_1, \dots, U_p) = \mathcal{I}_d,$$

and for any $x \in \mathbb{R}^d$, we can express x in terms of its block components as:

$$x = (x^{(1)}, \dots, x^{(p)})^T, \quad \text{where} \quad x^{(\ell)} = U_\ell^\top x \in \mathbb{R}^{d_\ell}, \quad \forall \ell \in \{1, \dots, p\},$$

and $x = \sum_{\ell=1}^p U_\ell x^{(\ell)}$. Then, memoryless cyclic coordinate descent can be defined as follows.

Algorithm 15 Memoryless Cyclic Coordinate Descent algorithms (MCCD)

- 1: **Input:** function f defined over $\mathcal{R}^n$ with p blocks, starting point $x_0 \in \mathcal{R}^n$, number of cycles K , and step-sizes $(\alpha_i)_{i=1,\dots,pK}$
 - 2: Define $N = pK$
 - 3: **for** $i = 1$ to N **do**
 - 4: Set $\ell = \text{mod}(i, p) + 1$
 - 5: $x_i = x_{i-1} - \frac{\alpha_i}{L_\ell} U_\ell \nabla^{(\ell)} f(x_{i-1})$
 - 6: **end for**
-

where $\nabla^{(\ell)} f(x) = U_\ell^\top \nabla f(x)$ is the partial gradient along the ℓ -block of coordinates. We optimize the step sizes $\{\alpha_i\}_{i=1}^{pK}$ of (MCCD)(α) over the functional class $\mathcal{F}_{0,\mathbf{L}}^{\text{coord}}(\mathbb{R}^d)$, under the assumptions of Setting INIT, where we only assume that $\|x_0 - x_*\|_{\mathbf{L}} \leq R$, with x_* an optimal point of the function (see Chapter 3, Equation 3.30). We choose as an initialization for the step sizes $\alpha_i^{(0)} = 1/L_{\ell(i)}$ for all $i \in \{1, \dots, pK\}$, where $\ell(i) = \text{mod}(i, p) + 1$. By the homogeneity result of Theorem 3.25, we can set $\mathbf{L} = (1, \dots, 1)$ and the initial distance to a minimizer to $R_i = 1$ without loss of generality. We further restrict our analysis to two blocks of coordinates ($p = 2$).

In Figure 5.4 we compare the numerical upper bounds we obtain using PEP for the classical cyclic coordinate descent algorithm with step sizes $\frac{1}{L_\ell}$ and for the cyclic coordinate descent with our optimized step sizes. We also provide numerical fits for these upper bounds. For the classical memoryless coordinate descent algorithm, we obtain:

$$f(x_N) - f(x_*) \leq \frac{1}{1.109N + 2.278} \|x_0 - x_*\|_{\mathbf{L}}^2.$$

For the coordinate descent algorithm with optimized step sizes, we have:

$$f(x_N) - f(x_*) \leq \frac{1}{1.087N^{1.145} + 2.226} \|x_0 - x_*\|_{\mathbf{L}}^2.$$

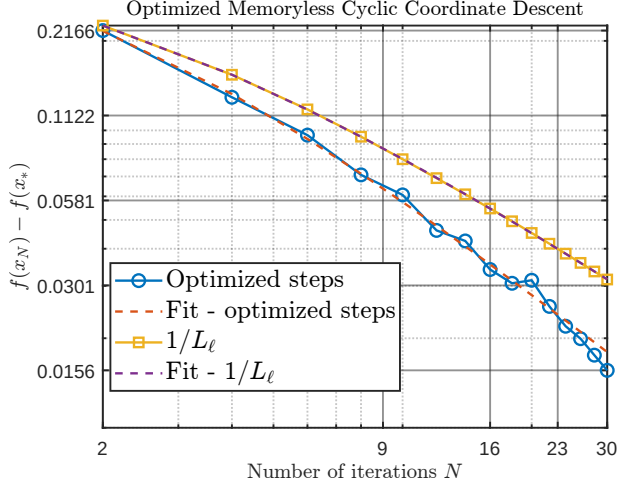


Fig. 5.4 Upper bounds on the worst-case convergence rate of memoryless cyclic coordinate descent with classical steps $\frac{1}{L_\ell}$ (yellow) and with our optimized step-sizes (blue).

The table below summarizes the optimized step sizes for memoryless cyclic coordinate descent for up to $N = 8$ iterations:

Number of Steps N	Step-Sizes α_i
2	[0.9761, 1.0637]
4	[0.9774, 1.5690, 0.9173, 1.0376]
6	[0.9016, 1.3699, 0.8865, 1.0576, 0.6993, 0.9311]
8	[0.9110, 1.5089, 0.8727, 1.4623, 1.1346, 1.2301, 0.7718, 1.0572]

Table 5.6 Optimized step-sizes for memoryless cyclic coordinate descent up to $N = 8$.

Remark 5.16. Note that the step sizes in Table 5.6 do not seem to follow any simple pattern that can be identified analytically. Unfortunately, this seems also to be the case for all the settings we study in the remainder of this section.

Conjecture 5.17. *Memoryless cyclic coordinate descent with optimized step sizes can achieve a faster convergence rate than $\mathcal{O}(1/N)$. For 2-blocks algorithms, our experiments suggest a rate of $\mathcal{O}(1/N^{1.145})$.*

We also apply our algorithm design method to the optimization of the step

sizes of a full fixed-step cyclic block coordinate-wise algorithm, where we allow the use of all past gradient information to perform the update, similarly to Nesterov’s acceleration. We place ourselves in the exact same setting as previously for the memoryless coordinate descent. In Figure 5.5, we compare the upper bounds obtain for both the optimized full cyclic coordinate descent and for the optimized memoryless gradient descent highlighting the benefits of using past gradient information. We also provide a numerical fit for the upper bound we have on the convergence of optimized full cyclic coordinate descent:

$$f(x_N) - f(x_*) \leq \frac{1}{0.6111N^{1.470} + 3.041} \|x_0 - x_*\|_{\mathbf{L}}^2.$$

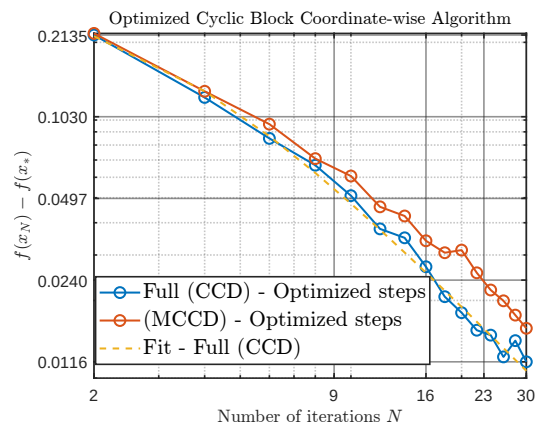


Fig. 5.5 Comparison of upper bounds on the worst-case convergence rate of our optimized full cyclic coordinate descent (blue) and optimized memoryless gradient descent (red).

Table 5.7 summarizes the optimized step sizes for the full fixed-step cyclic block coordinate-wise algorithm for up to $N = 6$ iterations.

Number of iterations N	Step-sizes $\alpha_{i,k}$					
2	0.8042					
	0.9999	1.0476				
4	0.9287					
	0.9919	1.3863				
	0.9956	1.4815	0.5634			
	0.9996	1.5760	1.0959	0.9439		
6	0.7878					
	1.0169		1.1648			
	1.0157	1.2016	0.2707			
	0.9964	1.3963	0.9591	1.1492		
	0.9984	1.4054	0.9559	1.1795	0.2679	
	0.9977	1.4673	1.0879	1.2667	0.9592	1.0037

Table 5.7 Optimized step-sizes for fixed-step full cyclic block coordinate-wise algorithms up to $N = 6$.

Conjecture 5.18. *Full cyclic coordinate descent with optimized step sizes can achieve a faster convergence rate than $\mathcal{O}(1/N)$. For 2-block algorithms, our experiments suggest a rate of $\mathcal{O}(1/N^{1.47})$.*

Conclusions. The numerical upper bounds in Figure 5.4, along with their corresponding fits, provide strong numerical evidence that accelerated convergence rates for cyclic coordinate descent on coordinate-wise smooth convex functions are achievable without memory (i.e., past gradient information). By accelerated, we mean here $\mathcal{O}(1/N^\alpha)$ with $\alpha > 1$. This aligns with recent results in [48, 45, 8] for the gradient descent algorithm on smooth convex functions. Also, the optimized step sizes for the first few iterations, summarized in Table 5.6, suggest that accelerating memory-less cyclic coordinate descent does not require long steps, unlike gradient descent on smooth convex functions [48, 44]. This observation is further supported by the optimized step sizes in Table 5.7 for block coordinate-wise algorithms that incorporate memory. The numerical results in Figure 5.5 indicate that using past gradient information, akin to Nesterov’s acceleration, helps to further accelerate the convergence of block coordinate-wise algorithms. However, our results suggest an approximate convergence rate of $\mathcal{O}(1/N^{1.47})$, which is significantly slower than the $\mathcal{O}(1/N^2)$ rate achieved by accelerated random block-coordinate descent algorithms. This supports the comparison made in Chapter 3 regarding the acceleration of random

versus cyclic block coordinate descent algorithms. Randomness appears to play a crucial role in accelerating block coordinate-wise algorithms.

Design of optimized inexact gradient descent algorithms for minimizing the residual gradient norm. We now consider the optimization of inexact gradient descent algorithms, where updates rely on inexact gradient estimates [102, 26]. We consider here relative inexactness on the gradients i.e we have only access to vectors d_i such that

$$\|d_i - \nabla f(x_i)\| \leq \epsilon \|\nabla f(x_i)\|.$$

This arises, for instance, in the training of machine learning models, where computationally cheaper gradient approximations, such as the mini-batch approximation for stochastic gradient methods, are used. Gradient inexactness can also result from noisy or missing data. We focus on the class of smooth convex functions and begin by analyzing standard memoryless inexact gradient descent algorithms:

Algorithm 16 Inexact gradient descent (IGD)

- 1: **Input:** Starting Point x_0 , number of iterations N , step-sizes α_i , tolerance ϵ
 - 2: **for** $i = 1$ to N **do**
 - 3: Compute d_i such that $\|d_i - \nabla f(x_i)\| \leq \epsilon \|\nabla f(x_i)\|$
 - 4: Update $x_{i+1} = x_i - \frac{\alpha_i}{L} d_i$
 - 5: **end for**
-

Note that inexact gradient descent algorithms can be easily handled within the PEP framework in a similar manner to gradient descent algorithms (see Chapter 2). We only need to introduce the extra variables d_i and the conditions $\|d_i - \nabla f(x_i)\| \leq \epsilon \|\nabla f(x_i)\|$. These conditions are readily SDP-representable using the same Gram matrix lifting technique as for gradient descent algorithms, but with the Gram matrix G defined as $G = P^\top P$, where $P = [g_0, \dots, g_N, d_0, \dots, d_{N-1}, x_0]$. For a more detailed explanation of the PEP formulation for inexact gradient descent, we refer the reader to [26]. Recently, the authors in [102] provided the first tight worst-case convergence analysis of inexact gradient descent for smooth convex functions.

Theorem 5.19. [102, Theorem 2.3] *Algorithm (IGD) applied to a convex L -smooth function with a constant stepsize $h \in]0, \frac{3}{2(1+\epsilon)}]$ and relative inexactness*

$\epsilon \in]0, 1[$, started from iterate x_0 , generates iterates satisfying

$$\frac{1}{L} \min_{k \in \{1, \dots, N\}} \|\nabla f(x_k)\|^2 \leq \frac{f(x_0) - f(x^*)}{\frac{1}{2} + Nh(1 - \epsilon)}.$$

For our numerical experiments, we choose the same performance criterion $\min_{k \in \{1, \dots, N\}} \|\nabla f(x_k)\|^2$ and initial condition $f(x_0) - f(x^*) \leq R$ as in Theorem 5.19, and we fix $L = 1$ and $R = 1$. We also apply our algorithm optimization method for several levels of inexactness: $\epsilon = 0.1, 0.3, 0.5$. We choose $\alpha_i^{(0)} = 1/L$ to initialize the step sizes.

In Figure 5.6 we provide numerical values of the worst-case convergence rate of our optimized algorithms, comparing them to the bound of Theorem 5.19. We also provide numerical fits of these upper bounds for our optimized inexact gradient methods.

For $\epsilon = 0.1$, we have:

$$\min_{i \in \{1, \dots, N\}} \|\nabla f(x)\|^2 \leq \frac{f(x_0) - f(x_*)}{1.211N^{1.116} + 0.5326}.$$

For $\epsilon = 0.3$, we have:

$$\min_{i \in \{1, \dots, N\}} \|\nabla f(x)\|^2 \leq \frac{f(x_0) - f(x_*)}{0.8077N^{1.076} + 0.5334}.$$

Finally, for $\epsilon = 0.5$, we have:

$$\min_{i \in \{1, \dots, N\}} \|\nabla f(x)\|^2 \leq \frac{f(x_0) - f(x_*)}{0.5414N^{1.018} + 0.5012}.$$

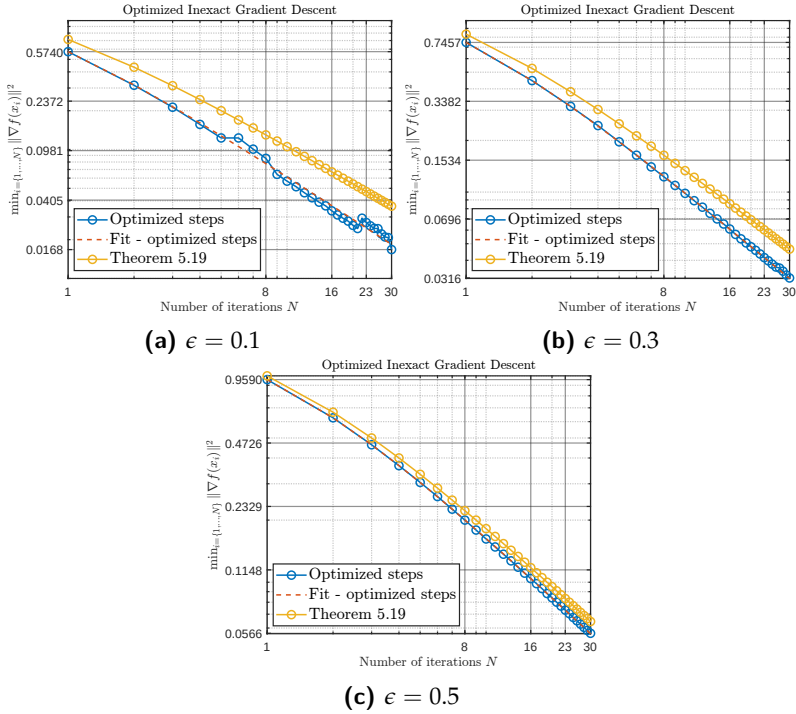


Fig. 5.6 Comparison of worst-case performance of our optimized inexact gradient methods compared to the inexact gradient descent with step-size $\alpha_i^{(0)} = 1/L$ for three different level of inexactness ϵ .

Conjecture 5.20. *Inexact gradient descent with optimized step sizes can achieve a faster convergence rate than $\mathcal{O}(1/N)$ for minimizing the squared gradient norm. We summarize the numerical convergence rates of our optimized inexact gradient descent algorithms in the following table:*

Level of Inexactness ϵ	Worst-Case Convergence Rate
0.1	$\mathcal{O}(1/N^{1.116})$
0.3	$\mathcal{O}(1/N^{1.076})$
0.5	$\mathcal{O}(1/N^{1.018})$

Table 5.8 Convergence rates of our optimized inexact gradient descent algorithms with respect to the level of inexactness ϵ .

We now provide the first few optimized step sizes obtained in this setting

for inexact gradient descent up to $N = 5$ iterations.

Number of iterations N	Optimized step-sizes α_i
1	[1.3885]
2	[1.6910, 1.3118]
3	[1.3885, 2.0748, 1.3054]
4	[1.3885, 2.5219, 1.4736, 1.2936]
5	[1.3885, 2.9444, 1.3236, 1.7528, 1.2874]

Table 5.9 Optimized step sizes for inexact gradient descent with a level of inexactness $\epsilon = 0.1$.

Number of iterations N	Optimized step-sizes α_i
1	[1.2232]
2	[1.4179, 1.1528]
3	[1.5595, 1.2743, 1.1011]
4	[1.2231, 1.8460, 1.2633, 1.0806]
5	[1.2231, 2.0715, 1.3373, 1.2380, 1.0535]

Table 5.10 Optimized step sizes for inexact gradient descent with a level of inexactness $\epsilon = 0.3$.

Number of iterations N	Optimized step-sizes α_i
1	[1.0998]
2	[1.2069, 1.0306]
3	[1.2935, 1.0927, 1.0081]
4	[1.3634, 1.1563, 1.0343, 1.0014]
5	[1.4152, 1.1876, 1.0828, 1.0095, 1.0001]

Table 5.11 Optimized step sizes for inexact gradient descent with a level of inexactness $\epsilon = 0.5$.

Conclusions. The numerical worst-case convergence rates of our optimized inexact gradient descent algorithms for levels of inexactness $\epsilon = 0.1, 0.3, 0.5$ in Figure 5.6, along with the corresponding numerical fits, indicate that achieving accelerated convergence rates compared to classical inexact gradient descent algorithms with constant step sizes (upper bound of Theorem 5.19) seems possible by carefully tuning the step sizes of the algorithms. Table 5.8, which summarizes the convergence rates achieved by our optimized algorithms with respect to the inexactness level, suggests that the

achievable acceleration depends highly on the level of inexactness. Indeed, it appears that higher acceleration is possible with lower levels of inexactness, whereas for high levels of inexactness, the acceleration tends to become minimal. Tables 5.9, 5.10, and 5.11 provide the optimized step sizes of our algorithms for the first 5 iterations. Note that the magnitude of these step sizes decreases with the level of inexactness. For a low level of inexactness ($\epsilon = 0.1$), some of the step sizes are unusually large (i.e., greater than $2/L$), similar to the step sizes of recent optimized gradient descent methods in [44, 48, 8]. For higher levels of noise ($\epsilon = 0.5$), our step sizes become more standard (i.e., between $1/L$ and $2/L$).

Design of full inexact gradient algorithms for minimizing the residual gradient norm. In an attempt to further accelerate the minimization of the residual gradient norm in the context of inexact optimization of smooth convex functions, we now consider full inexact gradient algorithms, meaning that we allow the use of all past inexact gradient information to perform the updates.

Algorithm 17 Full inexact gradient descent (FIGD)

- 1: **Input:** Starting Point x_0 , number of iterations N , step-sizes $\alpha_{i,k}$, tolerance ϵ
 - 2: **for** $i = 0$ to $N - 1$ **do**
 - 3: Compute d_i such that $\|d_i - \nabla f(x_i)\| \leq \epsilon \|\nabla f(x_i)\|$
 - 4: Update $x_i = x_0 - \sum_{k=0}^{i-1} \frac{\alpha_{i,k}}{L} d_k$
 - 5: **end for**
-

We place ourselves in the exact same setting as before and make the same assumptions as in the optimization of standard inexact gradient descent algorithms. In Figure 5.7, we provide numerical upper bounds on the worst-case convergence of our optimized (FIGD) and compare them to the bound for the unaccelerated inexact gradient descent (Theorem 5.19). We also provide numerical fits of these upper bounds for our optimized inexact gradient methods with different levels of inexactness.

For $\epsilon = 0.1$, we have:

$$\min_{i \in \{1, \dots, N\}} \|\nabla f(x)\|^2 \leq \frac{f(x_0) - f(x_*)}{0.9853N^{1.384} + 0.7563}.$$

For $\epsilon = 0.3$, we have:

$$\min_{i \in \{1, \dots, N\}} \|\nabla f(x_i)\|^2 \leq \frac{f(x_0) - f(x_*)}{0.6055N^{1.342} + 0.7373}.$$

Finally, for $\epsilon = 0.5$, we have:

$$\min_{i \in \{1, \dots, N\}} \|\nabla f(x_i)\|^2 \leq \frac{f(x_0) - f(x_*)}{0.4438N^{1.190} + 0.6005}.$$

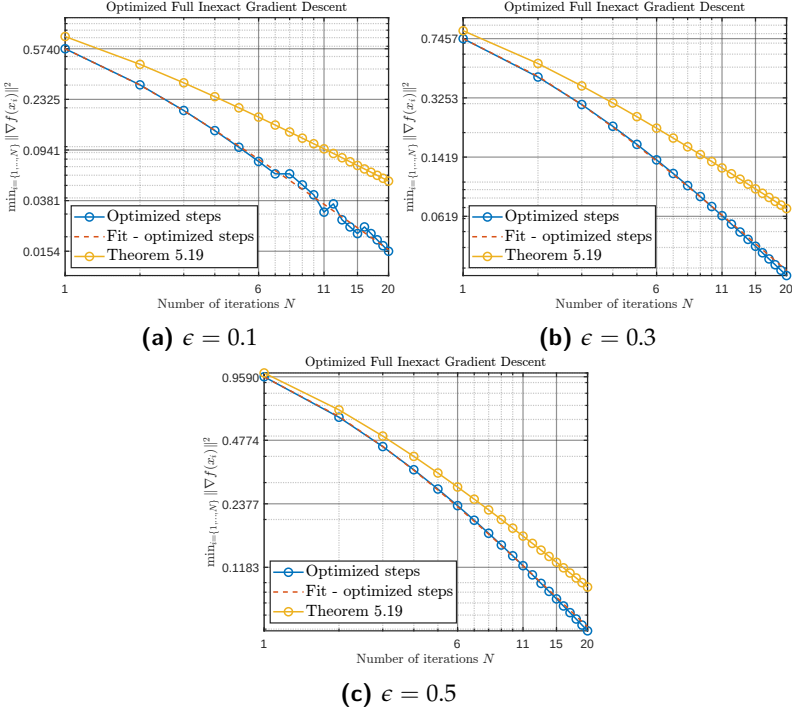


Fig. 5.7 Comparison of upper bound on the worst-case performance of our optimized full inexact gradient methods compared to the inexact gradient descent with constant step sizes (upper bound Theorem 5.19) for three different level of inexactness ϵ .

Conjecture 5.21. *Full Inexact descent with optimized step sizes can achieve a faster convergence rate than $\mathcal{O}(1/N)$ for minimizing the squared gradient norm. We summarize the numerical convergence rates of our optimized full inexact*

gradient descent algorithms in the following table:

Level of Inexactness ϵ	(FIGD)	(IGD)
0.1	$\mathcal{O}(1/N^{1.384})$	$\mathcal{O}(1/N^{1.116})$
0.3	$\mathcal{O}(1/N^{1.342})$	$\mathcal{O}(1/N^{1.076})$
0.5	$\mathcal{O}(1/N^{1.190})$	$\mathcal{O}(1/N^{1.018})$

Table 5.12 Comparison of the worst-case convergence rates of our optimized full inexact gradient descent and our optimized inexact gradient descent with respect to the level of inexactness ϵ .

We now provide the first few optimized step sizes we obtain in this setting for full inexact gradient descent.

Number of iterations N	Optimized step-sizes α_i			
1	1.3885			
2	1.6266 1.7272	1.4684		
3	1.7395 1.9982 2.0383	1.8119 1.9378	1.4563	
4	1.8057 2.1834 2.3135 2.3332	1.9871 2.3270 2.3783	1.8083 1.9305	1.4443

Table 5.13 Optimized step-sizes for full inexact gradient descent with a level of inexactness $\epsilon = 0.1$.

Number of iterations N	Optimized step-sizes α_i			
1	1.2232			
2	1.3487 1.4268	1.2232		
3	1.4568 1.5859 1.6189	1.4205 1.5280	1.1965	
4	1.4973 1.7059 1.7701 1.7876	1.5786 1.7565 1.8051	1.3749 1.4773	1.1804

Table 5.14 Optimized step-sizes for full inexact gradient descent with a level of inexactness $\epsilon = 0.3$.

Number of iterations N	Optimized step-sizes α_i			
1	1.0998			
2	1.1576 1.2159	1.0903		
3	1.2003 1.2822 1.3148	1.1481 1.2069	1.0839	
4	1.2370 1.3410 1.3876 1.4071	1.2081 1.3014 1.3404	1.1336 1.1895	1.0781

Table 5.15 Optimized step-sizes for full inexact gradient descent with a level of inexactness $\epsilon = 0.5$.

Conclusions. The numerical worst-case convergence rates in Figure 5.7, along with their numerical fits, indicate that the use of memory (past gradient information) indeed clearly helps accelerate the convergence of inexact gradient descent algorithms. However, Table 5.12, which summarizes the convergence rates achieved by our optimized algorithms, suggests that the achievable rates in the inexact setting are significantly slower than those achieved by optimal gradient descent methods for minimizing the residual

gradient norm of smooth convex functions, namely $\mathcal{O}(1/N^2)$ (see [56]). Furthermore, Tables 5.13, 5.14, and 5.15 seem to confirm our observations regarding the amplitude of optimized step sizes with respect to the level of inexactness (i.e., longer steps are useful for acceleration when the level of inexactness is low, but shorter steps are preferable when the level of inexactness is higher).

Design of inexact and full (i.e using past gradient information) inexact gradient descent algorithms for minimizing the functional accuracy.

Previously, we optimized the inexact gradient descent algorithm for minimizing the residual gradient norm on smooth convex functions to compare our results with the only known bound in the inexact setting. However, the residual gradient norm is not the standard performance criterion in convex optimization. Thanks to the flexibility of our PEP-based algorithm design framework, we now optimize inexact gradient descent algorithms for minimizing the functional error $f(x_N) - f(x_*)$, under the initial condition $\|x_0 - x_*\| \leq R$. As before, we fix $L = R = 1$ for our numerical experiments and initialize our optimization with the inexact gradient descent algorithm using a constant step size $\alpha_i^{(0)} = 1/L$. We then compare the performance of our optimized algorithms to this standard algorithm.

We begin by optimizing the step sizes of inexact gradient descent without memory and present in Figure 5.8 numerical worst-case convergence rates of our optimized algorithms and the standard inexact gradient descent with step sizes $\alpha_i^{(0)} = \frac{1}{L}$ for levels of inexactness $\epsilon = 0.1, 0.3, 0.5$. We also provide numerical fits for these convergence rates.

For $\epsilon = 0.1$, we have for the inexact gradient descent with constant step sizes $\alpha_i^{(0)} = 1/L$:

$$f(x_N) - f(x_*) \leq \frac{1}{3.623N + 1.976} \|x_0 - x_*\|^2$$

and for our optimized algorithm:

$$f(x_N) - f(x_*) \leq \frac{1}{4.519N^{1.171} + 2.452}.$$

For $\epsilon = 0.3$, we have for the inexact gradient descent with constant step

sizes $\alpha_i^{(0)} = 1/L$:

$$f(x_N) - f(x_*) \leq \frac{1}{2.976N^{0.9507} + 1.825} \|x_0 - x_*\|^2$$

and for our optimized algorithm:

$$f(x_N) - f(x_*) \leq \frac{1}{3.33N^{1.052} + 2.035}.$$

For $\epsilon = 0.5$, we have for the inexact gradient descent with constant step sizes $\alpha_i^{(0)} = 1/L$:

$$f(x_N) - f(x_*) \leq \frac{1}{2.207N^{0.8786} + 1.793} \|x_0 - x_*\|^2$$

and for our optimized algorithm:

$$f(x_N) - f(x_*) \leq \frac{1}{2.346N^{0.9291} + 1.814}.$$

Conjecture 5.22. *Inexact gradient descent with optimized step sizes can achieve an accelerated convergence rate compared to inexact gradient descent with step sizes $\frac{1}{L}$ for minimizing the functional accuracy. We summarize the worst-case convergence rates in the following table.*

Level of Inexactness ϵ	(IGD)	Optimized (IGD)
0.1	$\mathcal{O}(1/N)$	$\mathcal{O}(1/N^{1.171})$
0.3	$\mathcal{O}(1/N^{0.9507})$	$\mathcal{O}(1/N^{1.052})$
0.5	$\mathcal{O}(1/N^{0.8786})$	$\mathcal{O}(1/N^{0.9291})$

Table 5.16 Worst-case convergence rates for our optimized inexact gradient descent algorithms compared to inexact gradient descent with step-sizes $\alpha_i^{(0)} = 1$.

Note that for $\epsilon = 0.5$, even with optimized steps (IGD) cannot reach a $\mathcal{O}(1/N)$ rate.

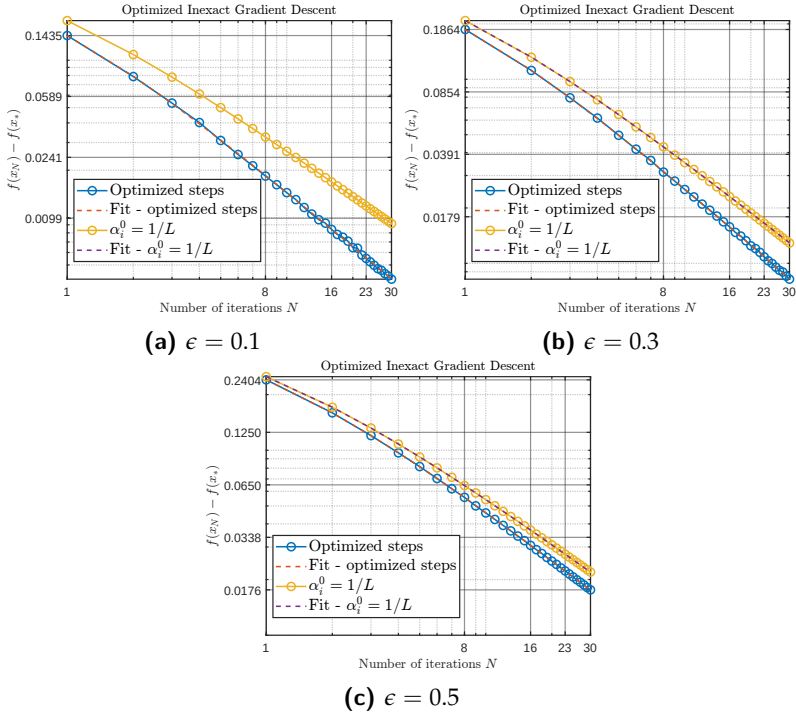


Fig. 5.8 Comparison of the worst-case performance of our optimized inexact gradient methods compared to the inexact gradient descent with constant step-sizes $\alpha_i^{(0)} = 1/L$ for three different level of inexactness ϵ .

In the next tables, we give the first few optimized step sizes we obtain for the different level of inexactness ϵ .

Number of iterations N	Optimized step-sizes α_i
1	[1.3885]
2	[1.3118, 1.6910]
3	[1.3005, 1.4753, 1.9326]
4	[1.2898, 1.4722, 1.5570, 2.1356]
5	[1.2828, 1.4695, 1.5573, 2.9044, 1.3885]

Table 5.17 Optimized step-sizes for inexact gradient descent with a level of inexactness $\epsilon = 0.1$.

Number of iterations N	Optimized step-sizes α_i
1	[1.2231]
2	[1.1585, 1.4205]
3	[1.1315, 1.2855, 1.5695]
4	[1.1065, 1.2733, 1.3449, 1.6881]
5	[1.0921, 1.2654, 1.3450, 2.0919, 1.2232]

Table 5.18 Optimized step-sizes for inexact gradient descent with a level of inexactness $\epsilon = 0.3$.

Number of iterations N	Optimized step-sizes α_i
1	[1.1115]
2	[1.0583, 1.2359]
3	[1.0252, 1.1508, 1.3277]
4	[1.0000, 1.1320, 1.1965, 1.3985]
5	[1.0000, 1.1157, 1.1922, 1.2260, 1.4456]

Table 5.19 Optimized step-sizes for inexact gradient descent with a level of inexactness $\epsilon = 0.5$.

We now optimize the full inexact gradient descent algorithm to minimize functional accuracy. Figure 5.9 presents numerical worst-case convergence rates for our optimized algorithms and compares them to those of inexact gradient descent with step sizes $\alpha_i^{(0)} = 1/L$. We also provide numerical fits for these convergence rates.

For $\epsilon = 0.1$:

$$f(x_N) - f(x_*) \leq \frac{1}{3.55N^{1.450} + 3.423} \|x_0 - x_*\|^2.$$

For $\epsilon = 0.3$:

$$f(x_N) - f(x_*) \leq \frac{1}{2.551N^{1.275} + 2.819} \|x_0 - x_*\|^2.$$

For $\epsilon = 0.5$:

$$f(x_N) - f(x_*) \leq \frac{1}{1.689N^{1.151} + 2.479} \|x_0 - x_*\|^2.$$

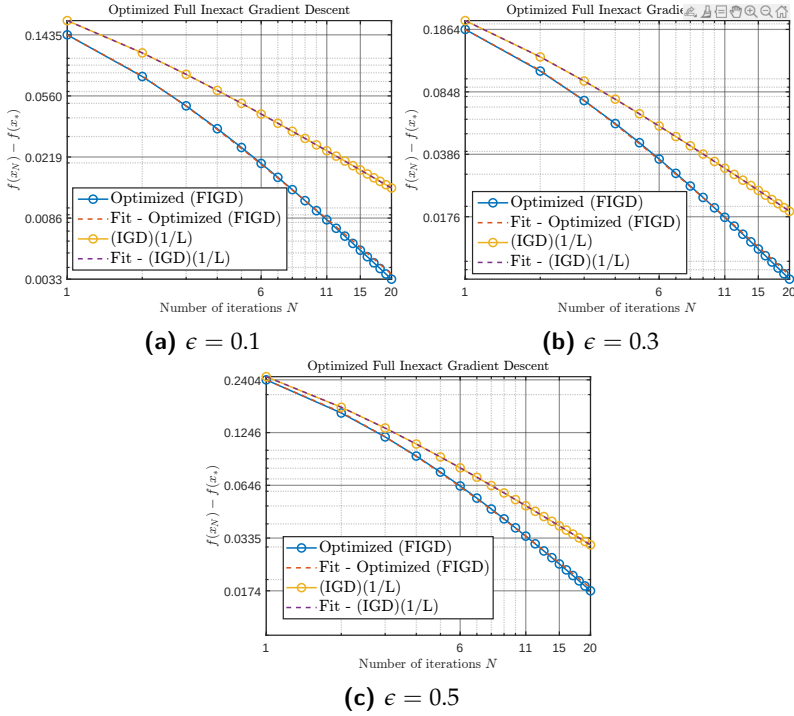


Fig. 5.9 Comparison of worst-case performance of our optimized full inexact gradient methods, (FIGD), compared to the inexact gradient descent with constant step-sizes $\alpha_i^{(0)} = 1/L$, (IGD), for three different level of inexactness ϵ .

Conjecture 5.23. *Full Inexact gradient descent with optimized step sizes can achieve an accelerated convergence rate compared to optimized inexact gradient descent and inexact gradient descent with step sizes $\frac{1}{L}$ for minimizing the functional accuracy. We summarize the worst-case convergence rates in the following table:*

Inexactness level ϵ	(IGD)	Optimized (IGD)	Optimized (FIGD)
0.1	$\mathcal{O}(1/N)$	$\mathcal{O}(1/N^{1.171})$	$\mathcal{O}(1/N^{1.450})$
0.3	$\mathcal{O}(1/N^{0.9507})$	$\mathcal{O}(1/N^{1.052})$	$\mathcal{O}(1/N^{1.275})$
0.5	$\mathcal{O}(1/N^{0.8786})$	$\mathcal{O}(1/N^{0.9291})$	$\mathcal{O}(1/N^{1.151})$

Table 5.20 Worst-case convergence rates of the optimized full inexact gradient descent algorithm (FIGD) compared to inexact gradient descent with step sizes $\alpha_i^{(0)} = 1/L$, (IGD) and optimized (IGD).

The next tables present the optimized step sizes for the first few iterations of our full inexact gradient descent algorithm.

Number of iterations N	Optimized step-sizes α_i			
1	[1.3885]			
2	1.4518	1.5547	1.6360	
3	1.4287	1.5538	1.7861	
	1.5960	2.0519	1.7615	
4	1.4090	1.5283	1.7596	
	1.5810	2.0951	1.9640	
	1.6024	2.2315	2.3559	1.8388

Table 5.21 Optimized step-sizes for full inexact gradient descent with a level of inexactness $\epsilon = 0.1$.

Number of iterations N	Optimized step-sizes α_i			
1	1.2231			
2	1.2188	1.2743	1.4028	
3	1.1730	1.2280	1.4489	
	1.2478	1.6111	1.4979	
4	1.1376	1.1549	1.4053	
	1.1497	1.5898	1.5802	
	1.1475	1.6685	1.8276	1.5552

Table 5.22 Optimized step-sizes for full inexact gradient descent with a level of inexactness $\epsilon = 0.3$.

Number of iterations N	Optimized step-sizes α_i			
1	1.1115			
2	1.0748 1.0965	1.2319		
3	1.0259 0.9999 0.9872	1.2343 1.3235	1.3000	
4	1.0049 0.9470 0.8832 0.8498	1.1882 1.2637 1.2954	1.3210 1.4642	1.3423

Table 5.23 Optimized step-sizes for full inexact gradient descent with a level of inexactness $\epsilon = 0.5$.

Conclusions. The numerical upper bounds in Figure 5.8 indicate that carefully tuning the step sizes in an inexact gradient descent algorithm accelerates convergence for minimizing functional accuracy. However, convergence remains slower than for minimizing the residual gradient norm. Moreover, the convergence speed, even for standard inexact gradient descent, strongly depends on the level of inexactness in the gradients. Tables 5.17, 5.18, and 5.19 provide the optimized step sizes for the first few iterations, reinforcing again the observations made for residual gradient norm minimization: the amplitude of the optimized step sizes decreases as inexactness increases. The numerical rates in Figure 5.9, along with the corresponding numerical fits, suggest that incorporating memory further accelerates the convergence of inexact gradient descent for functional accuracy minimization. However, our optimized methods remain significantly slower than optimal gradient methods in the exact minimization setting which have a $\mathcal{O}(1/N^2)$ rate of convergence. The optimized step sizes in Tables 5.21, 5.22, and 5.23 further confirm that optimized step sizes decrease as inexactness increases.

Design of optimized memoryless gradient descent and full gradient descent algorithms for L_β -Hölder-smooth convex minimization. We optimize fixed-step first-order algorithms to minimize the function accuracy $f(x_N) - f(x_*)$ for L_β -Hölder smooth convex functions, assuming $\|x_0 - x_*\| \leq R$. The convex PEP framework developed in Chapter 4 for

worst-case analysis of first-order algorithms on such functions does not fully align with our algorithmic framework for optimizing algorithms, as it includes a power cone constraint in addition to a semidefinite constraint. However, nonlinearity with respect to step sizes and other dual variables arises only in the semidefinite constraint, allowing the optimization of fixed-step first-order methods using our SLM approach to be handled similarly to other PEP frameworks where the algorithm design problem reduces to a purely nonlinear semidefinite program, with some added power cone constraints.

By the homogeneity result of Theorem 4.9, we set $L_\beta = 1$ and $R = 1$. We initialize the step sizes as $\alpha_i^{(0)} = 1/L$ and first optimize the steps of memoryless gradient descent (MGD). Figure 5.10 presents a comparison between numerical upper bounds on the worst-case convergence of our optimized memoryless gradient descent algorithms and the constant step-size memoryless gradient descent algorithm we proposed in Chapter 4 Equation (4.9) for different smoothness levels $\beta = 0.1, 0.5, 0.7$. This shows that our optimized algorithms exhibit faster convergence rates than the algorithm we proposed in Chapter 4. We also provide numerical fits for these upper bounds.

For $\beta = 0.1$:

$$f(x_N) - f(x_*) \leq \frac{1}{1.349N^{0.6069} + 0.8542} \|x_0 - x_*\|^{1+\beta}.$$

For $\beta = 0.5$:

$$f(x_N) - f(x_*) \leq \frac{1}{3.013N^{0.8056} + 0.9176} \|x_0 - x_*\|^{1+\beta}.$$

For $\beta = 0.7$:

$$f(x_N) - f(x_*) \leq \frac{1}{3.914N^{0.9485} + 1.352} \|x_0 - x_*\|^{1+\beta}.$$

Conjecture 5.24. *Memoryless gradient descent with optimized step sizes can achieve a faster convergence rate than (UGM) on Hölder-smooth convex functions of a fixed, known level of smoothness. The following table summarizes the convergence rates of our optimized memoryless gradient descent algorithms and compares them to the rates of the universal gradient method (UGM) from [73].*

Level of smoothness β	Optimized (MGD)	(UGD)
0.1	$\mathcal{O}(1/N^{0.6069})$	$\mathcal{O}(1/N^{0.55})$
0.5	$\mathcal{O}(1/N^{0.8056})$	$\mathcal{O}(1/N^{0.75})$
0.7	$\mathcal{O}(1/N^{0.9485})$	$\mathcal{O}(1/N^{0.85})$

Table 5.24 Worst-case convergence rates for our optimized memoryless gradient descent algorithms (MGD) compared to the universal gradient method (UGM) in [73] for different level of smoothness β .

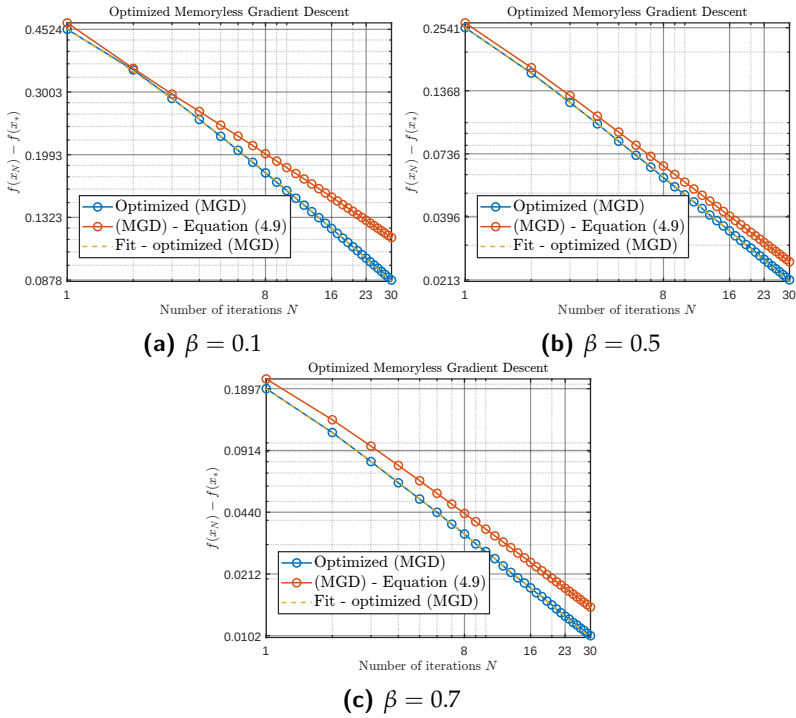


Fig. 5.10 Upper bounds on the worst-case convergence of optimized (MGD) for different level of smoothness β .

We give in the next tables, the first few optimized step sizes we obtain for different level of smoothness β .

Number of iterations N	Optimized step-sizes α_i
1	[0.9825]
2	[0.7579, 0.5377]
3	[0.6611, 0.5618, 0.3474]
4	[0.5922, 0.5733, 0.3880, 0.2515]
5	[0.5454, 0.5591, 0.4113, 0.3078, 0.1937]

Table 5.25 Optimized step-sizes for (MGD) and a level of smoothness $\beta = 0.1$.

Number of iterations N	Optimized step-sizes α_i
1	[1.0445]
2	[1.0186, 0.8234]
3	[0.9672, 0.9270, 0.6634]
4	[0.9206, 0.9423, 0.7818, 0.5723]
5	[0.7755, 1.0976, 0.7822, 0.6986, 0.5084]

Table 5.26 Optimized step-sizes for (MGD) and a level of smoothness $\beta = 0.5$.

Number of iterations N	Optimized step-sizes α_i
1	[1.1955]
2	[1.2495, 1.0470]
3	[0.9778, 1.5014, 0.9212]
4	[0.9454, 1.6057, 1.0584, 0.8517]
5	[0.9199, 1.6535, 1.0961, 1.0037, 0.8004]

Table 5.27 Optimized step-sizes for (MGD) and a level of smoothness $\beta = 0.7$.

We now optimize full gradient descent (FGD), allowing the use of past gradients as defined by (2.8), to achieve further acceleration in convergence rates. The next figure presents numerical upper bounds on the worst-case convergence of our optimized algorithms.

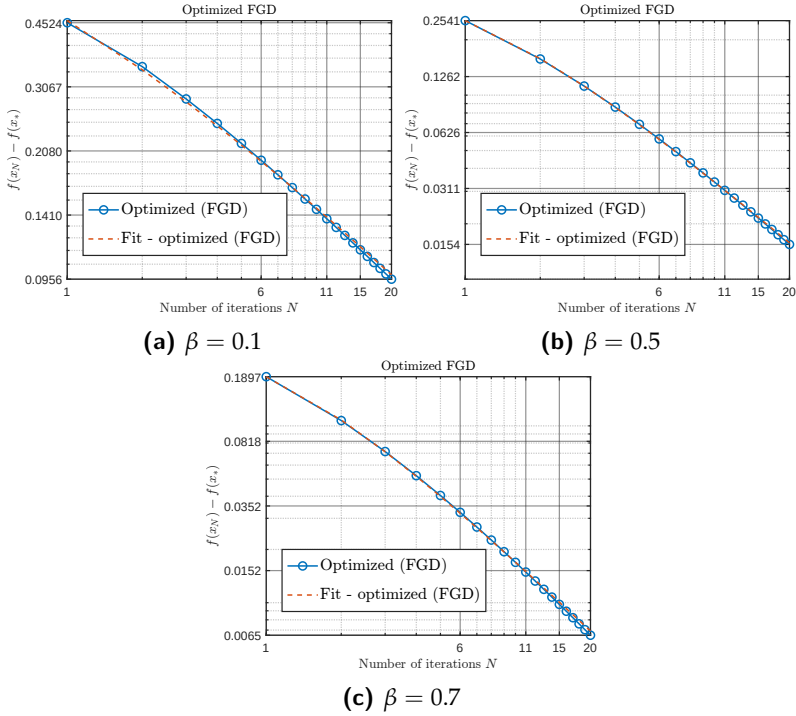


Fig. 5.11 Upper bounds on the worst-case convergence of optimized full gradient descent (FGD) methods for different level of smoothness β .

We provide numerical fits of the upper bounds of the previous figures. For $\beta = 0.1$ we have:

$$f(x_N) - f(x_*) \leq \frac{1}{1.332N^{0.65} + 0.85} \|x_0 - x_*\|^{1+\beta}.$$

For $\beta = 0.5$, we have:

$$f(x_N) - f(x_*) \leq \frac{1}{1.97N^{1.149} + 1.968} \|x_0 - x_*\|^{1+\beta}.$$

For $\beta = 0.7$, we have:

$$f(x_N) - f(x_*) \leq \frac{1}{2.632N^{1.331} + 2.643} \|x_0 - x_*\|^{1+\beta}.$$

In the next table, we summarize the convergence rates of our optimized

full gradient descent (FGD) algorithms and compare them to the optimal convergence rate achievable [69]. For instance, the optimal convergence rate on Hölder-smooth convex functions is achieved by the fast universal gradient method (FUGM) [73]. Table 5.28 suggests that our optimized (FGD) methods achieve the optimal rate for $\beta = 0.1$ but not for $\beta = 0.5, 0.7$.

Level of smoothness β	Optimized (FGD)	(FUGM)
0.1	$\mathcal{O}(1/N^{0.65})$	$\mathcal{O}(1/N^{0.65})$
0.5	$\mathcal{O}(1/N^{1.149})$	$\mathcal{O}(1/N^{1.25})$
0.7	$\mathcal{O}(1/N^{1.331})$	$\mathcal{O}(1/N^{1.55})$

Table 5.28 Worst-case convergence rates for our optimized (FGD) compared to the fast universal gradient method (FUGM) in [73] for different level of smoothness β .

Conjecture 5.25. *Our full gradient descent with optimized step sizes achieves suboptimal convergence compared to the fast universal gradient method.*

We give in the next tables, the numerical values of the optimized step-sizes we obtain for the first few iterations up to $N = 1$ and for different level of smoothness β .

Number of iterations N	Optimized step-sizes α_i			
1	0.9825			
2	0.7368 0.7549	0.5405		
3	0.6559 0.6659 0.6697	0.4385 0.5346	0.3801	
4	0.5858 0.5944 0.5990 0.6010	0.4043 0.5141 0.5643	0.3137 0.3951	0.2818

Table 5.29 Optimized step-sizes for full gradient descent (FGD) and a level of smoothness $\beta = 0.1$.

Number of iterations N	Optimized step-sizes α_i			
1	1.0445			
2	0.9599 1.0326	0.8848		
3	0.8917 0.9783 1.0088	0.8842 1.0427	0.7567	
4	0.8438 0.9247 0.9624 0.9784	0.8328 1.0286 1.1117	0.7859 0.9722	0.6645

Table 5.30 Optimized step-sizes for full gradient descent (FGD) and a level of smoothness $\beta = 0.5$.

Number of iterations N	Optimized step-sizes α_i			
1	1.1955			
2	1.1780 1.2735	1.1699		
3	1.1272 1.2449 1.2851	1.2360 1.4506	1.1040	
4	1.0910 1.2042 1.2554 1.2765	1.1931 1.4660 1.5785	1.2052 1.4815	1.0428

Table 5.31 Optimized step-sizes for full gradient descent (FGD) and a level of smoothness $\beta = 0.7$.

Conclusions. The upper bounds of Figure 5.10, along with the corresponding numerical fits, suggest that tuning the step sizes of a memoryless gradient descent algorithm accelerates the rate of convergence on Hölder-smooth convex functions compared to the universal gradient descent method (see Table 5.24). Furthermore, Tables 5.25, 5.26, 5.27 confirm the natural intuition that optimized step sizes for this setup should increase with the level of smoothness β , as the use of larger constant step sizes is common in smooth

convex optimization. In Figure 5.11, we present upper bounds on the worst-case convergence of optimized full gradient descent (allowing memory) and compare their convergence rates with the optimal one achieved by the fast universal gradient method for several levels of smoothness. It appears that the use of memory further accelerates convergence, but our optimized algorithms remain suboptimal compared to the fast universal gradient method which requires line-searches and is not a fixed-step method. Whether the optimal rate in this setting is achievable by a fixed-step method remains an open question. Tables 5.29, 5.30, 5.31 provide our optimized step sizes for the first few iterations and tend to confirm the observations we made above (for memoryless gradient descent) regarding optimized step sizes in the setting of Hölder-smooth convex optimization.

Design of cyclic gradient descent algorithms in the context of linear convergence. In the cases where we have linear convergence of gradient descent, our method can be used to design an optimized algorithm compared to the constant step-size gradient descent. Indeed, since we have linear convergence, we can fix a horizon N_* , optimize the step sizes for this horizon using our numerical methods, and then consider the cyclic algorithm that repeats the optimized step sizes found by our method. Then if our algorithm verifies after N_* iteration

$$\|x_{N_*} - x_*\|^2 \leq c(N_*)\|x_0 - x_*\|^2$$

we have that

$$\|x_{kN_*} - x_*\|^2 \leq c(N_*)^k \|x_0 - x_*\|^2, \forall k \in \mathbb{N}.$$

Algorithm 18 Cyclic gradient descent

- 1: **Input:** Starting point x_0 , number of steps N , length of cycles N_* , step-sizes $\alpha_1, \dots, \alpha_{N_*}$
 - 2: **for** $i = 1$ to N **do**
 - 3: Set $\ell = \text{mod}(i, N_*) + 1$
 - 4: $x_i = x_{i-1} - \alpha_\ell \nabla f(x_{i-1})$
 - 5: **end for**
-

Remark 5.26. Note that although we restrict our experiments to the class of smooth strongly convex functions, the methodology of designing a cyclic

algorithm remains valid in any scenario where linear convergence of first-order algorithms is guaranteed [66, 2].

In the next figure, we fix $N_* = 4$ and compare upper bounds on the worst-case performance of our optimized cyclic gradient descent algorithm to the classical gradient descent algorithm with the optimal constant step size $\frac{2}{L+\mu}$ on 1-smooth, 0.1-strongly convex functions.

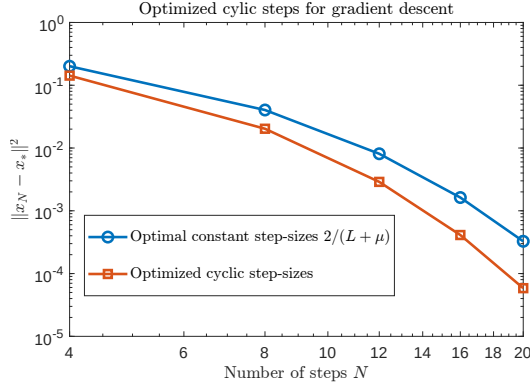


Fig. 5.12 Optimized cyclic steps for gradient descent for 1-smooth 0.1-strongly convex functions.

Conclusions. Our optimized cyclic steps improve the worst-case performance of gradient descent on smooth strongly convex functions. To quantify the performance gap, we provide numerical convergence bounds for both methods. For the optimal constant step sizes $\frac{2}{L+\mu}$, we have the worst-case performance:

$$\|x_{4N} - x_*\|^2 \leq 0.2008^N \|x_0 - x_*\|^2, \quad \forall N \in \mathbb{N}.$$

For gradient descent with our cyclic (with cycle length $N_* = 4$) optimized steps, we have the worst-case performance bound:

$$\|x_{4N} - x_*\|^2 \leq 0.14239^N \|x_0 - x_*\|^2, \quad \forall N \in \mathbb{N}.$$

This provides numerical evidence that our optimized cyclic step-size approach improves the worst-case performance of gradient descent when it converges linearly. Moreover, it is a useful method to trade off between the computational cost of optimizing the steps and the performance of the algorithm. Indeed, larger cycle lengths N_* improve performance but increase

the computational cost of optimizing the step sizes.

5.6 Conclusion

In this chapter, we showed how to formulate the problem of designing fixed-step first-order algorithms in a very generic manner, thanks to the use of the PEP framework. We then provided several methods to optimize first-order algorithms, including a successive linearization method that is competitive with the BNB-PEP method in [48] on the benchmark task of optimizing memoryless gradient descent on smooth convex functions. Furthermore, we provided numerical evidence indicating the computational efficiency of our method compared to the BNB-PEP approach. We then presented numerically optimized fixed-step first-order algorithms in a range of different settings, showing improved worst-case performance compared to classical fixed-step first-order algorithms available for these optimization settings. Our results suggest that recent findings on the accelerated convergence of first-order algorithms without the use of Nesterov's acceleration over smooth convex functions can be extended to other settings.

Open research directions include finding analytical forms for the algorithms presented in this chapter, developing more scalable methods for the automatic design of first-order algorithms, as our methods are computationally limited by the cost of solving large-scale SDPs, and extending the design of algorithms using PEP to different classes of algorithms, including, for instance, higher-order methods, decentralized algorithms, and adaptive methods.

6

Conclusion

In this thesis, we developed several extensions of the Performance Estimation Problem (PEP) framework for the automatic worst-case analysis and design of first-order methods. In the next section, we summarize the main contributions of this Thesis before discussing some interesting future perspective on the analysis and optimal design of first-order methods.

6.1 Summary of the Contributions

In Chapter 3, we extended the PEP framework for the worst-case analysis of block coordinate-wise descent algorithms (BCD) and provided a convex formulation of the PEP for the subclass of fixed-step block coordinate-wise algorithms and alternating minimization. We applied this framework to provide new insights into the behavior of several BCD algorithms. In particular, we provided improved numerical bounds for the cyclic coordinate descent (CCD) and alternating minimization (AM) algorithms and provided insights into the evolution of the worst-case of (CCD) with respect to the number of blocks, as well as the evolution of the optimal constant step size for (CCD) with respect to the number of blocks and cycles. Additionally, we compared the worst-case performance of cyclic accelerated coordinate descent (CACD) to its random version and observed a slower convergence rate, indicating the significance of randomness in accelerating block coordinate descent algorithms. We demonstrated the convergence of these algorithms

under the simplified constraints of Setting INIT, as opposed to the typically employed stronger assumptions of Setting ALL. Furthermore, we formally established several results concerning the behavior of (CCD), including scale invariance with respect to the block coordinate-wise smoothness constants, an improved descent lemma for 2-block (CCD) with a corresponding upper bound on the convergence for the residual gradient norm, and a lower bound on its worst-case performance for the function value accuracy criterion that is equal to the number of blocks multiplied by the worst-case performance of full gradient descent on smooth convex functions.

In Chapter 4, we extended the PEP framework for the analysis of first-order algorithms over the functional class of Hölder-smooth convex functions $\mathcal{F}_{0,L\beta}^\beta(\mathbb{R}^d)$. We provided necessary $\mathcal{F}_{0,L\beta}^\beta(\mathbb{R}^d)$ -interpolation conditions and used these conditions to derive a convex PEP formulation for the subclass of fixed-step first-order algorithms over this class of functions. We proposed a simple fixed-step first-order algorithm whose step sizes interpolate between the step sizes of the classical gradient descent algorithm with constant step size $\frac{1}{L}$ for L -smooth convex functions and the subgradient method with the decreasing step sizes $\frac{R}{L\sqrt{N}+1}$ for nonsmooth convex functions with difference of gradients bounded by L . We provided numerical upper bounds on the worst-case convergence of our algorithm, suggesting sublinear convergence over the functional classes $\mathcal{F}_{0,L\beta}^\beta(\mathbb{R}^d)$. However, compared to the performance of universal gradient methods, our algorithm exhibits suboptimal convergence for low levels of smoothness β . Building on this, we were able to find better numerical step sizes for gradient descent on these classes of functions using the methods we developed in Chapter 5.

In Chapter 5, we showed how to formulate the problem of designing fixed-step first-order algorithms in a very generic manner, thanks to the use of the PEP framework. We then provided several methods to optimize first-order algorithms, including a successive linearization method that is competitive with the BNB-PEP method in [48] on the benchmark task of optimizing memoryless gradient descent on smooth convex functions. Furthermore, we provided numerical evidence indicating the computational efficiency of our method compared to the BNB-PEP approach. We then presented numerically optimized fixed-step first-order algorithms in a range of different settings, showing improved worst-case performance compared to classical fixed-step first-order algorithms available for these optimization settings.

Our results suggest that recent findings on the accelerated convergence of first-order algorithms without the use of Nesterov's acceleration over smooth convex functions can be extended to other settings, namely coordinate descent algorithms, inexact gradient descent, and gradient descent on Hölder-smooth convex functions. Furthermore, we showed that using memory (past gradient information), similar to Nesterov's acceleration, can further accelerate the convergence of algorithms where no known acceleration schemes exist. This is particularly the case for cyclic coordinate descent and inexact gradient descent. Moreover, when the convergence of gradient descent is linear, we proposed a framework to design optimized algorithms in a computationally efficient manner by considering a cyclic algorithm, optimizing the step sizes for a cycle, and then using them as the step sizes of our algorithm with cyclic step sizes. Then, choosing the length of the cycle (and thus the number of step sizes to optimize) trades off between the speed of convergence and the computational cost of the step-size optimization process.

6.2 Future Perspectives

Analysis of optimization algorithms In some of our PEP formulations, the interpolation conditions for the functional class are only proven to be necessary but not sufficient. This is, for instance, the case for coordinate-wise smooth convex functions and Hölder-smooth convex functions. This possibly introduces conservatism in the PEP framework, which then only yields valid upper bounds on the worst-case performance and not the exact worst-case of the algorithm. Finding sufficient interpolation conditions is a bottleneck for PEP, as it is not an easy problem to solve, and there is no standard procedure to derive interpolation conditions from the definition of a functional class. Moreover, there is no guarantee that these interpolation conditions can be integrated into a convex formulation for the PEP. An interesting line of future research would be to develop a principled approach to derive interpolation conditions, balancing the trade-off between their sufficiency and simplicity. This means finding interpolation conditions that yield the tightest upper bound on the worst-case convergence of the algorithm while still being integrable into a convex PEP formulation. In [87] the authors propose an approach to strengthen interpolation conditions, but which does not guarantee that the resulting interpolation conditions can be integrated into an SDP. Being able to quantify a priori the conservatism induced by using only necessary conditions would also be an important

step in understanding the significance of having tight representations of functional classes. For instance, in the scope of this thesis, it would be interesting to assess how conservative our upper bounds are for the convergence of block coordinate-wise algorithms and gradient descent algorithms on Hölder-smooth convex functions. The upper bounds we provide for our fixed-step first-order algorithm on Hölder-smooth convex functions suggest that its performance is suboptimal compared to universal gradient methods. However, it remains unclear whether this suboptimality arises from insufficient interpolation conditions or is an intrinsic property of fixed-step first-order algorithms on this class of functions.

Furthermore, some of the remaining limitations of PEP are linked to the type of analysis it can perform and the class of algorithms it can efficiently analyze in a convex manner. As of now, PEP can only perform worst-case analysis on optimization algorithms, which is often conservative compared to their practical performance. Developing a novel, less conservative, automatic method to evaluate the performance of optimization algorithms—providing an average estimate of an algorithm’s performance over a class of functions—would be an interesting extension to PEP. Convex formulations of PEPs are typically only possible for fixed-step first-order methods. A natural question is whether this can be generalized to other classes of algorithms, such as higher-order methods. Moreover, PEP analysis requires prior knowledge of the total number of steps performed by the algorithm and the exact problem parameters for instance the value of the smoothness constant L , and the initial distance to a minimizer R , which are generally unknown in practice. Developing a convex PEP framework that does not require knowledge of these parameters would be a challenging yet important research direction.

Design of Algorithms In many instances, the worst-case functions of fixed-step first-order algorithms are finite in number and one-dimensional. This is, for instance, the case for gradient descent on smooth (strongly) convex functions, for which optimized algorithms were designed by balancing gradient descent’s performance across its one-dimensional worst-case functions [8, 45]. Generalizing this approach to algorithm design, for example, by considering only one-dimensional functions (which translates into a rank constraint on the SDP variable of the PEP) or by finding a way to automatically identify the worst-case functions and optimizing the algorithm on these functions, could yield a highly efficient method for optimizing fixed-

step first-order algorithms. Furthermore, optimizing algorithms over a finite set of parameterized functions is of interest for applications such as machine learning and control theory. In relation to the previous paragraph, once an efficient PEP formulation is developed for adaptive algorithms—without requiring prior knowledge of the number of iterations or of the problem parameters—extending this framework to design fully adaptive algorithms could have significant practical implications. Similarly, extending PEP to design optimal higher-order methods or optimal algorithms in average over a functional class remain interesting open research directions. Finally, in many optimization settings, information-theoretic lower bounds are missing. This is the case, for instance, in the recently studied setting of memoryless first-order algorithms for smooth convex optimization. Developing lower bounds specific to this class of algorithms would be highly valuable in guiding future research toward optimal memoryless first-order algorithms.

Bibliography

- [1] H. Abbaszadehpeivasti, E. de Klerk, and M. Zamani. The exact worst-case convergence rate of the gradient method with fixed step lengths for l -smooth functions. *Optimization Letters*, 16:1649–1661, 2022.
- [2] H. Abbaszadehpeivasti, E. de Klerk, and M. Zamani. Conditions for linear convergence of the gradient method for non-convex optimization. *Optimization Letters*, 17:1105–1125, 2023.
- [3] Hadi Abbaszadehpeivasti, Etienne de Klerk, and Moslem Zamani. Convergence rate analysis of randomized and cyclic coordinate descent for convex optimization through semidefinite programming. *Applied Set-Valued Analysis and Optimization*, 5(2):141–153, 2023.
- [4] Hadi Abbaszadehpeivasti, Etienne de Klerk, and Moslem Zamani. On the rate of convergence of the difference-of-convex algorithm (dca). *Journal of Optimization Theory and Applications*, pages 1–22, 2023.
- [5] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research (JMLR)*, 18(1):8194–8244, 2017.
- [6] Zeyuan Allen-Zhu, Zheng Qu, Peter Richtarik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1110–1119. PMLR, 20–22 Jun 2016.
- [7] Jason M. Altschuler and Pablo A. Parrilo. Acceleration by stepsize hedging i: Multi-step descent and the silver stepsize schedule, 2023.

- [8] Jason M. Altschuler and Pablo A. Parrilo. Acceleration by stepsize hedging: Silver stepsize schedule for smooth convex optimization. *Mathematical Programming*, 2024.
- [9] MOSEK ApS. *MOSEK Optimizer API for Python*. MOSEK ApS, 2024. Version 10.1.
- [10] D. Azagra, E. Le Gruyer, and C. Mudarra. Explicit formulas for $c^{1,1}$ and $c_{\text{conv}}^{1,\omega}$ extensions of 1-jets in hilbert and superreflexive spaces. *Journal of Functional Analysis*, 274:3003–3032, 2018.
- [11] Mathieu Barré, Adrien B. Taylor, and Francis Bach. Principled analyses and design of first-order methods with inexact proximal operators. *Mathematical Programming*, 201(1-2):185–230, 2023.
- [12] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, Berlin, 2011.
- [13] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.
- [14] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.
- [15] J. Frédéric Bonnans and Alexander Shapiro. *Perturbation Analysis of Optimization Problems*. Springer Series in Operations Research and Financial Engineering. Springer, 2000.
- [16] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.
- [17] Nizar Bousselmi, Julien M. Hendrickx, and François Glineur. Interpolation conditions for linear operators and applications to performance estimation problems. *SIAM Journal on Optimization*, 34(3):3033–3063, 2024.
- [18] S. Boyd and L. El Ghaoui. Alternating minimization algorithms for bilinear matrix inequalities. In *Proceedings of the 1994 American Control Conference*, pages 687–691. IEEE, 1994.

- [19] Kristian Bredies. A forward–backward splitting algorithm for the minimization of non-smooth convex functionals in banach space. *Inverse Problems*, 25(1):015005, 2008.
- [20] Augustin-Louis Cauchy. Méthode générale pour la résolution des systèmes d’équations simultanées. *Comptes Rendus de l’Académie des Sciences*, 25:536–538, 1847.
- [21] Sebastien Colla and Julien M. Hendrickx. Automated worst-case performance analysis of decentralized gradient descent. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2627–2633, 2021.
- [22] Sebastien Colla and Julien M. Hendrickx. Automatic performance estimation for decentralized optimization. *IEEE Transactions on Automatic Control*, 68(12):7136–7150, 2023.
- [23] Rafael Correa and Héctor Ramírez Cabrera. A global algorithm for nonlinear semidefinite programming. *SIAM Journal on Optimization*, 15(1):303–318, 2004.
- [24] Antoine Daccache. Performance estimation of the gradient method with fixed arbitrary step sizes. Master’s thesis, Université Catholique de Louvain, 2019.
- [25] Shuvomoy Das Gupta, Robert M. Freund, X. Andy Sun, and Adrien B. Taylor. Nonlinear conjugate gradient methods: worst-case convergence rates via computer-assisted analyses. *Mathematical Programming*, 2024.
- [26] Etienne De Klerk, François Glineur, and Adrien B Taylor. On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions. *Optimization Letters*, 11:1185–1199, 2017.
- [27] Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146:37–75, 2014.
- [28] Jelena Diakonikolas and Lorenzo Orecchia. Alternating randomized block coordinate descent. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1224–1232. PMLR, 10–15 Jul 2018.

- [29] R.A. Dragomir, A.B. Taylor, A. d’Aspremont, et al. Optimal complexity and certification of bregman first-order methods. *Mathematical Programming*, 194:41–83, 2022.
- [30] Yoel Drori. The exact information-based complexity of smooth convex minimization. *Journal of Complexity*, 39:1–16, 2017.
- [31] Yoel Drori and Adrien Taylor. On the oracle complexity of smooth strongly convex minimization. *Journal of Complexity*, 68, 2022.
- [32] Yoel Drori and Adrien B. Taylor. Efficient first-order methods for convex minimization: A constructive approach. *Mathematical Programming*, 184:183–220, 2020.
- [33] Yoel Drori and Marc Teboulle. Performance of first-order methods for smooth convex minimization: A novel approach. *Mathematical Programming*, 145(1–2):451–482, 2014.
- [34] Diego Eloi. Worst-case functions for the gradient method with fixed variable step sizes. Master’s thesis, Université Catholique de Louvain, 2022.
- [35] B. Fares, D. Noll, and P. Apkarian. Robust control via sequential semidefinite programming. *SIAM Journal on Control and Optimization*, 40(6):1791–1820, 2002.
- [36] Olivier Fercoq and Peter Richtárik. Accelerated, parallel and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.
- [37] Alexander V. Gasnikov and Yurii E. Nesterov. Universal method for stochastic composite optimization problems. *Computational Mathematics and Mathematical Physics*, 58(1):48–64, 2018.
- [38] Eduard Gorbunov, Adrien Taylor, Samuel Horváth, and Gauthier Gidel. Convergence of proximal point and extragradient-based methods beyond monotonicity: the case of negative comonotonicity. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11614–11641. PMLR, 23–29 Jul 2023.
- [39] Baptiste Goujaud, Aymeric Dieuleveut, and Adrien Taylor. On fundamental proof structures in first-order optimization. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 3023–3030, 2023.

- [40] Baptiste Goujaud, Charles Moucer, François Glineur, et al. Pepit: computer-assisted worst-case analyses of first-order optimization methods in python. *Mathematical Programming Computation*, 16:337–367, 2024.
- [41] Baptiste Goujaud, Damien Scieur, Aymeric Dieuleveut, Adrien B. Taylor, and Fabian Pedregosa. Super-acceleration with cyclical step-sizes. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3028–3065, 2022.
- [42] Baptiste Goujaud, Adrien B. Taylor, and Aymeric Dieuleveut. Optimal first-order methods for convex functions with a quadratic upper bound. *arXiv preprint arXiv:2205.15033*, 2022.
- [43] Benjamin Grimmer. General hölder smooth convergence rates follow from specialized rates assuming growth bounds. *Journal of Optimization Theory and Applications*, 197(1):51–70, 2023.
- [44] Benjamin Grimmer. Provably faster gradient descent via long steps. *SIAM Journal on Optimization*, 34(3):2588–2608, 2024.
- [45] Benjamin Grimmer, Kevin Shu, and Alex L. Wang. Composing optimized stepsize schedules for gradient descent, 2024.
- [46] Benjamin Grimmer, Kevin Shu, and Alex L. Wang. A strengthened conjecture on the minimax optimal constant stepsize for gradient descent, 2024.
- [47] Charles Guille-Escuret, Adam Ibrahim, Baptiste Goujaud, and Ioannis Mitliagkas. Gradient descent is optimal under lower restricted secant inequality and upper error bound. In *Advances in Neural Information Processing Systems*, volume 35, pages 24893–24904, 2022.
- [48] Shuvomoy Das Gupta, Bart P.G. Van Parys, and Ernest K. Ryu. Branch-and-bound performance estimation programming: A unified methodology for constructing optimal optimization methods. *Mathematical Programming*, 204:567–639, 2024.
- [49] Mert Gurbuzbalaban, Asuman Ozdaglar, Pablo A. Parrilo, and Nuri Vanli. When cyclic coordinate descent outperforms randomized coordinate descent. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

- [50] Filip Hanzely and Peter Richtárik. Accelerated coordinate descent with arbitrary sampling and best rates for minibatches. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pages 304–312, 2019.
- [51] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms I and II*, volume 305 and 306 of *Grundlehren der mathematischen Wissenschaften*. Springer, Heidelberg, 1996. Two volumes, 2nd printing.
- [52] Mingyi Hong, Xiangfeng Wang, Meisam Razaviyayn, and Zhi-Quan Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, 163(1-2):85–114, 2017.
- [53] Uijeong Jang, Shuvomoy Das Gupta, and Ernest K. Ryu. Computer-assisted design of accelerated composite optimization methods: Optista. *arXiv preprint arXiv:2305.15704*, 2023.
- [54] Yassine Kamri, Julien M. Hendrickx, and François Glineur. On the worst-case analysis of cyclic coordinate-wise algorithms on smooth convex functions. In *2023 European Control Conference (ECC)*, pages 1–8, 2023.
- [55] Donghwan Kim and Jeffrey A. Fessler. Optimized first-order methods for smooth convex minimization. *Mathematical Programming*, 159:81–107, 2016.
- [56] Donghwan Kim and Jeffrey A. Fessler. Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions. *Journal of Optimization Theory and Applications*, 188(1):192–219, 2021.
- [57] Jungbin Kim. A proof of the exact convergence rate of gradient descent, 2025.
- [58] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.
- [59] Tianjiao Li and Guanghui Lan. A simple uniformly optimal method without line search for convex optimization, 2024.

- [60] Xiaojing Li, Tianyang Zhao, Raman Arora, Han Liu, and Mingyi Hong. An improved convergence analysis of cyclic block coordinate descent-type methods for strongly convex minimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 491–499. PMLR, 2016.
- [61] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3384–3392, 2015.
- [62] Q. Lin, Z. Lu, and L. Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015.
- [63] Alan Luner and Benjamin Grimmer. Performance estimation for smooth and strongly convex sets. *arXiv preprint arXiv:2410.14811*, 2024.
- [64] Renato D. Monteiro and Benar F. Svaiter. An accelerated hybrid proximal extragradient method for convex optimization. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.
- [65] Céline Moucer, Adrien Taylor, and Francis Bach. A systematic approach to lyapunov analyses of continuous-time models in convex optimization. *SIAM Journal on Optimization*, 33(3):1558–1586, 2023.
- [66] Ion Necoara, Yurii Nesterov, and François Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Mathematical Programming*, 175:69–107, 2019.
- [67] A. Nemirovskii and Yu. Nesterov. Optimal methods for smooth convex optimization. *Zhurnal Vychislitel’noi Matematiki i Matematicheskoi Fiziki*, 25(3):356–369, 1985.
- [68] A. S. Nemirovsky and B. T. Polyak. The chebyshev iterative method for solving linear equations. *USSR Computational Mathematics and Mathematical Physics*, 24(4):189–194, 1984.
- [69] Arkadii Nemirovsky and David Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, 1983.

- [70] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.
- [71] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2004.
- [72] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
- [73] Yurii Nesterov. Universal gradient methods for convex optimization problems. *Mathematical Programming*, 152:381–404, 2015.
- [74] Yurii Nesterov. Implementable tensor methods in unconstrained convex optimization. *Mathematical Programming*, 186:157–183, 2021.
- [75] Yurii Nesterov and Sebastian U. Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017.
- [76] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.
- [77] Daniel P. Palomar and Yonina C. Eldar. *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, 2010.
- [78] Jisun Park and Ernest K Ryu. Exact optimal accelerated complexity for fixed-point iterations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17420–17457. PMLR, 17–23 Jul 2022.
- [79] Jongmin Park and Ernest K. Ryu. Exact optimal accelerated complexity for fixed-point iterations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17420–17457. PMLR, July 2022.
- [80] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
- [81] Vinit Ranjan, Jisun Park, Stefano Gualandi, Andrea Lodi, and Bartolomeo Stellato. Exact verification of first-order methods via mixed-integer linear programming. *arXiv e-prints*, December 2024.

- [82] James B. Rawlings and David Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.
- [83] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, 1996.
- [84] Anton Rodomanov, Ali Kavis, Yongtao Wu, Kimon Antonakopoulos, and Volkan Cevher. Universal gradient methods for stochastic convex optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 42620–42646. PMLR, 21–27 Jul 2024.
- [85] Teodor Rotaru, François Glineur, and Panagiotis Patrinos. Exact worst-case convergence rates of gradient descent: a complete analysis for all constant stepsizes over nonconvex and convex functions, 2024.
- [86] Anne Rubbens and Julien M. Hendrickx. A constraint-based approach to function interpolation, with application to performance estimation for weakly convex optimisation, 2024.
- [87] Anne Rubbens, Julien M. Hendrickx, and Adrien Taylor. A constructive approach to strengthen algebraic descriptions of function and operator classes, 2025.
- [88] A. Saha and A. Tewari. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.
- [89] Bryan Van Scoy, Randy A. Freeman, and Kevin M. Lynch. The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Systems Letters*, 2(1):49–54, 2018.
- [90] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 64–72, 2014.
- [91] Z. Shi and R. Liu. Better worst-case complexity analysis of the block coordinate descent method for large scale machine learning. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 889–892. IEEE, 2017.

- [92] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. MIT Press, 2012.
- [93] Ruoyu Sun and Mingyi Hong. Improved iteration complexity bounds of cyclic block coordinate descent for convex problems. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [94] Adrien Taylor and Francis Bach. Stochastic first-order methods: Non-asymptotic and computer-aided analyses via potential functions. In *Proceedings of the 32nd Conference on Learning Theory (COLT)*, pages 2934–2992, 2019.
- [95] Adrien Taylor and Yoel Drori. An optimal gradient method for smooth strongly convex minimization. *Mathematical Programming*, 199:557–594, 2023.
- [96] Adrien B. Taylor. *Convex Interpolation and Performance Estimation of First-order Methods for Convex Optimization*. PhD thesis, Université catholique de Louvain, 2017.
- [97] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, 161(1-2):307–345, 2017.
- [98] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. Performance estimation toolbox (pesto): Automated worst-case analysis of first-order optimization methods. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1278–1283, 2017.
- [99] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. Exact worst-case convergence rates of the proximal gradient method for composite convex minimization. *Journal of Optimization Theory and Applications*, 178, Aug 2018.
- [100] Marc Teboulle and Yakov Vaisbourd. An elementary approach to tight worst-case complexity analysis of gradient-based methods. *Mathematical Programming*, 201(1-2):63–96, 2023.
- [101] Onur Toker and Hitay Ozbay. On the NP-hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output

- feedback. In *1995 American Control Conference (ACC)*, pages 2525–2526. IEEE, 1995.
- [102] Pierre Vernimmen and François Glineur. Convergence analysis of an inexact gradient method on smooth convex functions. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgium, 2024.
- [103] Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.
- [104] Stephen J. Wright and Ching-Pei Lee. Analyzing random permutations for cyclic coordinate descent. *Mathematics of Computation*, 89(325):2217–2248, January 2020.
- [105] Maryam Yashtini. On the global convergence rate of the gradient descent method for functions with Hölder continuous gradients. *Optimization Letters*, 10(7):1361–1370, 2016.
- [106] Moslem Zamani, Hadi Abbaszadehpeivasti, and Etienne de Klerk. The exact worst-case convergence rate of the alternating direction method of multipliers. *Mathematical Programming*, pages 1–34, 2023.
- [107] Moslem Zamani and François Glineur. Exact convergence rate of the last iterate in subgradient methods, 2023.
- [108] Zehao Zhang and Rujun Jiang. Accelerated gradient descent by concatenation of stepsize schedules, 2024.
- [109] Constantin Zălinescu. *Convex Analysis in General Vector Spaces*. World Scientific, Singapore, 2002.