2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

On Fundamental Proof Structures in First-Order Optimization

Baptiste Goujaud and Aymeric Dieuleveut and Adrien Taylor

***Abstract*— First-order optimization methods have attracted a lot of attention due to their practical success in many applications, including in machine learning. Obtaining convergence guarantees and worst-case performance certificates for first-order methods have become crucial for understanding ingredients underlying efficient methods and for developing new ones. However, obtaining, verifying, and proving such guarantees is often a tedious task. Therefore, a few approaches were proposed for rendering this task more systematic, and even partially automated. In addition to helping researchers finding convergence proofs, these tools provide insights on the general structures of such proofs. We aim at presenting those structures, showing how to build convergence guarantees for first-order optimization methods.**

I. Introduction

In recent years, there has been a significant surge in the interest surrounding first-order optimization methods, primarily driven by their remarkable efficiency on a number of applications, notably within the field of machine learning (see e.g., [5]). Theoretical foundations for those methods played a crucial role in this success, e.g., by enabling the development of momentum-type methods (see e.g., [29], [26]). Formally, we consider the optimization problem

$$x_\star \triangleq \arg\min_{x\in\mathbb{R}^d} f(x) \tag{OPT}$$

where f belongs to a set $\mathcal{F}$ (often referred to as a "class of functions", e.g., the set of convex functions, the set of strongly-convex and smooth functions, or the set of quadratic convex functions, etc.). Classical first-order optimization methods for solving this problem include *gradient descent* (GD) [7], *Nesterov accelerated gradient method* (NAG) [26], and the *heavy-ball method* (HB) [29].

In this context, a key question is to obtain a priori performance guarantees for an iterative algorithm $\mathcal{A}$ (i.e., $\mathcal{A}$ is a rule for generating sequences of approximations $(x_t)_{t\leqslant T}$ to the minimizers of a certain function f) when the function f to be minimized belongs to a set $\mathcal{F}$. The most popular framework for such analyses of optimization algorithms is that of *worst-case analyses*, see, e.g., [26], [15], [6], [16], [8]. Given an algorithm $\mathcal{A}$, the *worst-case analysis* framework consists in finding guarantees that hold for *every* function of the class.

In other words, we aim at evaluating the worst-case accuracy of $\mathcal{A}$ over the functions of the class $\mathcal{F}$ after a given number of iterations T. For doing so, there are many different possible notions of *accuracy* (or performance) which we denote by $P(f,(x_t)_{t\leqslant T})$ and that we aim at minimizing. Letting x_T be the output of an algorithm, common examples of such metrics include the distance of the last iterate to an optimum $\|x_T-x_\star\|$, the function value accuracy of the last iterate $f(x_T)-f(x_\star)$, or its gradient norm $\|\nabla f(x_T)\|$. Usually, x_T can be arbitrarily bad just by choosing x_0 arbitrarily far away from the optimizer $x_\star$. Therefore, we usually need to assume x_0 to be not too bad, such as $x_0\in\mathcal{N}(x_\star)$ where $\mathcal{N}(x_\star)$ can be any fixed set (that we call a "neighborhood" of the optimizer $x_\star$) and depends on $x_\star$. Common examples of such neighborhood are balls around the optimizer $\{x|\|x-x_\star\|\leqslant R\}$ or the set $\{x|f(x)-f_\star\leqslant R\}$.

The smallest upper bound on $P(f,(x_t)_{t\leqslant T})$ that holds for any dimension $d\geqslant 1$, for any function $f\in\mathcal{F}$, for any starting point $x_0\in\mathcal{N}(x_\star)\subset\mathbb{R}^d$, and for any $(x_t)_{t\leqslant T}$ generated by $\mathcal{A}$ applied on f from x_0, is the optimal value to the problem of computing the worst-case:

$$\left|\begin{array}{ll} \underset{\substack{f\in\mathcal{F},d\geqslant 1\\ (x_t)_{t\leqslant T}\in\left(\mathbb{R}^d\right)^{T+1}}}{\text{maximize}} & P(f,(x_t)_{t\leqslant T}) \\ \text{subject to} & \begin{cases} x_0\in\mathcal{N}(x_\star) \\ (x_t)_{t\leqslant T}=\mathcal{A}(f,T,x_0) \end{cases} \end{array}\right. \tag{$\mathcal{P}$}$$

In the black-box model, iterative algorithms gather information about f through so-called *oracles*, which we denote by $\mathcal{O}^{(f)}$. Classical oracles used in first-order optimization are gradient evaluations $\mathcal{O}^{(f)}(x)=\nabla f(x)$ and approximate gradients $\mathcal{O}^{(f)}(x)\approx\nabla f(x)$ (e.g., stochastic gradients), but also proximal operators (see, e.g. [9]), etc. At step $t\in[\![1,T]\!]$, $\mathcal{A}$ collects oracles on the previous iterates $(\mathcal{O}^{(f)}(x_s))_{s\leqslant t-1}$ and outputs x_t based on those information through the update function A_t as $x_t=A_t((x_s,\mathcal{O}^{(f)}(x_s))_{s<t})$.

Notation. For readability purposes, all notation used throughout this paper are summarized as follows.

Notation	Corresponding object
$\mathcal{F}$	Class of functions (generic form)
f	Objective function
$x_\star$	Optimal point
x_0	Initial iterate
$\mathcal{O}^{(f)}$	Generic oracle applied on f
$\mathcal{A}$	Algorithm (generic form)
$(x_t)_{t\leqslant T}$	Sequence of iterates generated by $\mathcal{A}$, i.e. $(x_t)_{t\leqslant T}=\mathcal{A}(f,T,x_0)$
$(A_t)_{1\leqslant t\leqslant T}$	Update function of the algorithm $\mathcal{A}$, i.e. $\forall t, x_t=A_t((x_s,\mathcal{O}^f(x_s))_{s<t})$
T	Total number of iterations
t	Current iteration index
$\mathcal{F}_{\mu,L}$	Class of L-smooth and μ-strongly-convex functions ($0\leqslant\mu\leqslant L$)
$\mathcal{Q}_{\mu,L}$	Class of L-smooth and μ-strongly convex quadratic functions ($0\leqslant\mu\leqslant L$)
$(V_t)_t$	Lyapunov sequence
F,G	Linearization variables (after SDP lifting)
$P(f,(x_t)_{t\leqslant T})$	Performance metric

Baptiste Goujaud, CMAP, Ecole Polytechnique, Institut Polytechnique de Paris, Route de Saclay, 91120 Palaiseau, France. baptiste.goujaud@gmail.com.

Aymeric Dieuleveut, CMAP, Ecole Polytechnique, Institut Polytechnique de Paris, Route de Saclay, 91120 Palaiseau, France. aymeric.dieuleveut@polytechnique.edu.

Adrien Taylor, INRIA Paris, 2 Rue Simone IFF, 75012 Paris. adrien.taylor@inria.fr.

Outline. In Section II, we discuss two ways of characterizing classes of functions and detail the main cases for which we can solve ($\mathcal{P}$). In Section III, we discuss an alternative way of describing the algorithm $\mathcal{A}$ simplifying the resolution of ($\mathcal{P}$). Section IV outlines a systematic approach for acquiring proofs of worst-case performance certificates and delves into their underlying structures. We further elaborate on how this structure can be exploited for extending the applicability range of the worst-case guarantees. Among others, we show how the properties of these proofs allow building algorithms. Finally, Section VI provides a natural approach for discovering Lyapunov sequences.

II. From explicit to implicit classes of functions

This section describes two ways of specifying a class of functions as part of the worst-case analysis of a given algorithm. We describe two different methods to approach and solve ($\mathcal{P}$) depending on the ways $\mathcal{F}$ is specified. More specifically, we focus on two specific classes of functions to illustrate our explanations, namely L-smooth μ-strongly convex quadratic functions (notation $\mathcal{Q}_{\mu,L}$) and L-smooth μ-strongly convex functions (notation $\mathcal{F}_{\mu,L}$).

A. Convex quadratic optimization

First-order optimization methods were extensively studied in the context of minimizing quadratic convex functions. Such functions can be described **explicitly** as

$$f(x) \triangleq \frac{1}{2}(x-x_\star)^T H(x-x_\star) + f_\star, \tag{1}$$

where H is the symmetric positive semi-definite Hessian of f, $x_\star$ its optimizer and $f_\star$ its minimal value. This expression allows to explicitly compute the gradient $\nabla f(x) = H(x - x_\star)$, and first-order optimization methods can be expressed through polynomials due to the following property (e.g., [21, Prop.4.1]).

Proposition 2.1: Let $f \in \mathcal{Q}_{0,\infty}$ and $x_0 \in \mathbb{R}^d$. It holds that

$$x_{t+1} \in x_0 + \operatorname{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}, \tag{2}$$

if and only if there exists a sequence of polynomials $(P_t)_{t\in\mathbb{N}}$, each of degree at most 1 more than the highest degree of all previous polynomials and P_0 of degree 0 (hence the degree of P_t is at most t), such that

$$\forall\, t \quad x_t - x_\star = P_t(H)(x_0 - x_\star), \quad P_t(0) = 1\,. \tag{3}$$

In this context, ($\mathcal{P}$) can be solved by solving a polynomial problem of the form $\max_H \|P_t(H)\|$ where H is a symmetric matrix verifying some conditions (e.g. $\mu I \preceq H \preceq LI$ when $f \in \mathcal{Q}_{\mu,L}$). This link between first-order algorithms and polynomials has been used by [19] for discovering the Chebyshev method and by [29] for the "heavy-ball" method, still used nowadays far beyond quadratic optimization (e.g. in stochastic optimization of neural networks [33]). This property has also been exploited more recently for obtaining new algorithms with provable guarantees on quadratic functions (see e.g., [17], [30], [16], [28], [31], [4], [21], [23], [10]).

B. Infinite-dimensional spaces of functions

As opposed to previous sections, many classes of functions are described *implicitly* as regions of infinite-dimensional spaces of functions. In other words, such functions are defined by sets of inequalities. This section deals with the analyses of such classes. This is due to the fact the set of all functions of the class are not described by a finite number of parameters, but rather by constraints (inequalities). Studying ($\mathcal{P}$) for classes that are defined **implicitly** through sets of constraints appears to be much less natural. In this situation, ($\mathcal{P}$) is often referred to as a *performance estimation problem (PEP)* [14], [36], [34]. This tool primarily relies on two crucial components: interpolation conditions and SDP lifting.

Interpolation conditions. We remark that the description of the algorithm and the objective of ($\mathcal{P}$) both only depend on the oracle values of f on the iterates $(x_t)_{t\leqslant T}$. We introduce the variables $(\mathcal{O}_t)_{t\leqslant T}$. The constraint $f \in \mathcal{F}$ must be replaced by the constraint that there exists at least one element $f \in \mathcal{F}$ such that $(\mathcal{O}^{(f)}(x_t))_{t\leqslant T} = (\mathcal{O}_t)_{t\leqslant T}$ ($\mathcal{O}_t$ is a reachable value for $\mathcal{O}^{(f)}(x_t)$, when $f \in \mathcal{F}$). As an example, f_t and g_t are potential values of respectively $f(x_t)$ and $\nabla f(x_t)$. Formally, we define the equivalence relation $\sim_{(\mathcal{P})}$ as $f_1 \sim_{(\mathcal{P})} f_2$ if and only if $\forall t \in [\![0,T]\!] \cup \{\star\}, \mathcal{O}^{(f_1)}(x_t) = \mathcal{O}^{(f_2)}(x_t)$. Since the only information $\mathcal{A}$ gathers on f is the oracle outputs at the iterates x_t, two functions coming from the same equivalence class both produce feasible points of ($\mathcal{P}$) with the same objective value. In other words, those two functions are undistinguishable using only the information available to $\mathcal{A}$. We can therefore rewrite ($\mathcal{P}$) in terms of $(\mathcal{O}_t)_{t\leqslant T} \in \mathcal{F}/\sim_{(\mathcal{P})}$ instead of $f \in \mathcal{F}$, so that the set of optimization variables now lives in finite dimension. This constraint is referred to as *interpolation conditions*.

Example 2.2 (First-order algorithm on $\mathcal{F}_{\mu,L}$): Let $L \geqslant \mu > 0$ two positive real numbers. A function f is L-smooth and μ-strongly-convex when f is continuously differentiable and verifies the two inequalities:

$$f(x) \leqslant f(y) + \langle \nabla f(y), x-y\rangle + \frac{L}{2}\|x-y\|^2, \tag{4}$$

$$f(x) \geqslant f(y) + \langle \nabla f(y), x-y\rangle + \frac{\mu}{2}\|x-y\|^2, \tag{5}$$

for all x, y and where ∇f denotes the gradient of f.

Studying a first-order algorithm (i.e. an algorithm based on the oracle $\mathcal{O}^{(f)} \triangleq (\nabla f, f)$) on the class $\mathcal{F}_{\mu,L}$ appears to be challenging at first sight due to the infinite number of parameters needed for describing $\mathcal{F}_{\mu,L}$. However, [36, Theorem 4] provides interpolation conditions for the class $\mathcal{F}_{\mu,L}$ of L-smooth μ-strongly-convex functions and enables an exact study of the worst-case of several algorithms on this class of functions:

$$\begin{aligned}\forall i,j,\ \ f_i \geqslant & f_j + \langle g_j, x_i - x_j\rangle + \tfrac{1}{2L}\|g_i - g_j\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\|x_i - \tfrac{1}{L}g_i - x_j + \tfrac{1}{L}g_j\|^2.\end{aligned} \tag{IC}$$

Indeed, in this case, ($\mathcal{P}$) can be written in finite

dimension as

$$\begin{array}{l}\underset{\substack{d\geqslant 1,(x_t)_{t\leqslant T}\in(\mathbb{R}^d)^{T+1},x_\star\in\mathbb{R}^d,\\ (g_t,f_t)_{t\leqslant T}\in(\mathbb{R}^d\times\mathbb{R})^{T+1}}}{\text{maximize}} \quad P((x_t,g_t,f_t)_{t\leqslant T}) \\ \text{s.t.} \begin{cases} x_0\in\mathcal{N}(x_\star) \\ \forall t\leqslant T,\ x_t = A_t((x_s,\mathcal{O}^f(x_s))_{s<t}) \\ \forall i,j,\ f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i-g_j\|^2 \\ \qquad\qquad + \frac{\mu}{2(1-\mu/L)}\|x_i - \frac{1}{L}g_i - x_j + \frac{1}{L}g_j\|^2. \end{cases}\end{array}$$

SDP lifting. In many cases (see, e.g, Example 2.2, and [34, Theorem 3.5]), interpolation conditions are written in terms of quadratic and bilinear expressions of x_t and g_t and linear expressions of f_t. Because of the quadratic dependency in x_t and g_t, this problem is generally non-convex. *SDP lifting* can convexify this problem if all other parts of this problem also contain only quadratic expressions of x_t and g_t. For example, classical choices for $P(f,(x_t)_{t\leqslant T})$ are $\|x_T - x_\star\|^2$, $f(x_T) - f(x_\star)$, or $\|\nabla f(x_T)\|^2$. Similarly, a classical choice for $x_0\in\mathcal{N}(x_\star)$ is $\|x_0-x_\star\|^2\leqslant R^2$ for some radius $R>0$. Finally, the updates $(A_t)_t$ of the algorithm $\mathcal{A}$ are often of the form

$$x_t = A_t((x_s,\nabla f(x_s),f(x_s))_{s\leqslant t-1}) = x_0 - \sum_{s=0}^{t-1}\gamma_s^{(t)}\nabla f(x_s) \tag{6}$$

for some sequence of scalars $(\gamma_s^{(t)})_{s\in[\![0,t-1]\!]}$. Substituting x_t for $t\geqslant 1$ in the problem by their corresponding expressions given by (6) preserves the above observation: the dependency of ($\mathcal{P}$) in $(x_t,g_t)_{t\leqslant T}$ is exclusively quadratic. Actually, in this specific case, all occurrences of $(x_t)_{t\geqslant 1}$ have been replaced by linear combinations of x_0 and $(g_t)_{t\leqslant T}$. *SDP lifting* consists in introducing the Gram matrix G of $(x_0-x_\star,(g_t)_{t\leqslant T})$. This way, all quadratic expressions of $(x_t,g_t)_{t\leqslant T}$ are linear combinations of the entries of G. We also introduce the vector F storing the values $(f_t-f_\star)_{t\leqslant T}$.

Finally ($\mathcal{P}$) is rewritten with linear objective and constraints only as well as an SDP constraint $G\succeq 0$.

$$\begin{array}{ll}\underset{F,G\succeq 0}{\text{maximize}} & \langle F, v_P\rangle + \langle G, M_P\rangle \\ \text{subject to} & \begin{cases} \langle F, v_I\rangle + \langle G, M_I\rangle \leqslant R^2 \\ \forall k,\ \left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle \leqslant 0 \end{cases}\end{array}$$

Vectors $(v_P, v_I, (v_{\mathcal{F}}^{(k)})_k)$ and matrices $(M_P, M_I, (M_{\mathcal{F}}^{(k)})_k)$ are constants depending on the algorithm $\mathcal{A}$, the class $\mathcal{F}$, and the performance metric P under consideration. More specifically, indices P, I and $\mathcal{F}$ respectively correspond to the performance metric, the initialization constraint and the class interpolation conditions. The algorithm is directly encoded in the fact that G does not contain inner product with $(x_t)_{t\geqslant 1}$. As an example, to express $\|x_T-x_\star\|^2$ in terms of G, one needs to actually choose M with $\langle G, M\rangle = \|x_0 - x_\star - \sum_{s=0}^{T-1}\gamma_s^{(T)}\nabla f(x_s)\|^2$.

Key conditions. The above procedure generally works under the following conditions:

- $\mathcal{A}$ is a first-order algorithm whose updates $(A_t)_t$ can be expressed linearly in terms of observed gradients;
- The interpolation constraints of the class of functions $\mathcal{F}$ are known and expressible linearly in F and G;
- The performance metric as well as the initial condition are also expressible linearly in terms of F and G.

Many pairs of function class and algorithm meet the right conditions and have been studied using the PEP framework. Tools in Matlab [35] and Python [20] have been implemented to automate this task and provide worst-case guarantees. Many examples of usages are listed in the corresponding documentations.

III. From explicit to implicit algorithms

So far, we only considered explicit algorithms of the form (6). Note that, just as for classes of functions, algorithms can be expressed implicitly via sets of (in)equalities. This is the case for line-search based algorithms. Indeed, the step-size associated with line-search is not uniform over the problem class, therefore algorithms containing line-search update cannot be written as (6), and therefore do not meet the key conditions mentioned in the previous section. A relaxation of the gradient descent with exact line-search has been proposed in [11]. Since this algorithm cannot be written as (6) with pre-determined $\gamma_s^{(t)}$, we cannot specify $(x_t)_{t\geqslant 1}$ in terms of x_0 and $(g_t)_{t\leqslant T}$. Therefore, all vectors $(x_t,g_t)_{t\leqslant T}$ must be considered as linearly independent. For this problem, G is the Gram matrix of all $(x_t,g_t)_{t\leqslant T}$.

Therefore, the algorithm is not totally encoded in $v_P, M_P, v_I, M_I, v_{\mathcal{F}}$ and $M_{\mathcal{F}}$ anymore and must be specified by new constraints. In particular, the updates of gradient descent with line-search verify that

$$\langle g_{t+1}, g_t\rangle = 0 \tag{7}$$
$$\langle g_{t+1}, x_{t+1} - x_t\rangle = 0 \tag{8}$$

As for all the other elements of ($\mathcal{P}$), those constraints only involve quadratic terms of $(x_t,g_t)_{t\leqslant T}$ and can therefore be expressed linearly in terms of G, parametrized by the vectors $(v_{\mathcal{A}}^{(l)})_l$ and matrices $(M_{\mathcal{A}}^{(l)})_l$. This time, ($\mathcal{P}$) writes

$$\begin{array}{ll}\underset{F,G\succeq 0}{\text{maximize}} & \langle F, v_P\rangle + \langle G, M_P\rangle \\ \text{subject to} & \begin{cases} \langle F, v_I\rangle + \langle G, M_I\rangle & \leqslant R^2 \\ \forall k,\ \left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle & \leqslant 0 \\ \forall l,\ \left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle & \leqslant 0 \end{cases}\end{array}$$

(PEP-primal)

IV. Proof structures in first-order optimization

There is an extensive literature on first-order optimization, offering a broad range of possibly advanced worst-case guarantees and their associated proofs. In the previous sections, we saw conditions under which the problem of computing worst-case guarantees was tractable. In this section, we detail how to obtain proofs from PEPs and what we can conclude on the general structure of proofs in first-order optimization.

A. Obtaining proofs with PEPs

Thanks to interpolation conditions and SDP lifting, ($\mathcal{P}$) rewrites as a convex optimization problem. We consider the dual of the problem. Let's then introduce the Lagrangian multipliers τ, $(\lambda_{\mathcal{F}}^{(k)})_k$, $(\lambda_{\mathcal{A}}^{(l)})_l$ associated to the constraints of (PEP-primal).

$$\left|\begin{array}{ll} \underset{F,G\succeq 0}{\text{maximize}} & \langle F, v_P\rangle + \langle G, M_P\rangle \\ \text{subject to} & \left\{\begin{array}{rll} \langle F, v_I\rangle + \langle G, M_I\rangle & \leqslant R^2 & : \tau \\ \forall k, \left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle & \leqslant 0 & : \lambda_{\mathcal{F}}^{(k)} \\ \forall l, \left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle & \leqslant 0 & : \lambda_{\mathcal{A}}^{(l)} \end{array}\right. \end{array}\right.$$

The Lagrangian then writes

$$\begin{aligned} \mathcal{L} \triangleq\ & \langle F, v_P\rangle + \langle G, M_P\rangle - \tau\left[\langle F, v_I\rangle + \langle G, M_I\rangle - R^2\right] \\ & - \sum_k \lambda_{\mathcal{F}}^{(k)}\left[\left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle\right] \\ & - \sum_l \lambda_{\mathcal{A}}^{(l)}\left[\left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle\right] \\ =& \tau R^2 + \left\langle F, v_P - \tau v_I - \sum_k \lambda_{\mathcal{F}}^{(k)} v_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} v_{\mathcal{A}}^{(l)}\right\rangle \\ & + \left\langle G, M_P - \tau M_I - \sum_k \lambda_{\mathcal{F}}^{(k)} M_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} M_{\mathcal{A}}^{(l)}\right\rangle \end{aligned}$$

The dual is obtained by maximizing over the primal variables:

$$\left|\begin{array}{ll} \underset{\tau, \lambda_{\mathcal{F}}^{(k)}, \lambda_{\mathcal{A}}^{(l)} \geqslant 0}{\text{minimize}} & \tau R^2 \\ \text{s.t.} & \left\{\begin{array}{rl} v_P - \tau v_I - \sum_k \lambda_{\mathcal{F}}^{(k)} v_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} v_{\mathcal{A}}^{(l)} & = 0 \\ M_P - \tau M_I - \sum_k \lambda_{\mathcal{F}}^{(k)} M_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} M_{\mathcal{A}}^{(l)} & \preceq 0 \end{array}\right. \end{array}\right. \quad \text{(PEP-dual)}$$

For any feasible primal F, G and feasible dual $\tau, (\lambda_{\mathcal{F}}^{(k)})_k, (\lambda_{\mathcal{A}}^{(l)})_l$, we know the objective of the dual is larger than the Lagrangian value, that is:

$$\begin{aligned} & \underbrace{\langle F, v_P\rangle + \langle G, M_P\rangle}_{\text{Performance metric}} - \tau \underbrace{\left[\langle F, v_I\rangle + \langle G, M_I\rangle\right]}_{\text{Initialization}} \\ & \leqslant \sum_k \lambda_{\mathcal{F}}^{(k)} \underbrace{\left[\left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle\right]}_{\text{Class constraint}} \\ & \quad + \sum_l \lambda_{\mathcal{A}}^{(l)} \underbrace{\left[\left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle\right]}_{\text{Algorithm constraint}} \\ & \leqslant 0. \end{aligned} \quad \text{(Generic proof)}$$

In words, the proof of a worst-case guarantee is obtained by linearly combining all available constraints, with coefficients that are the dual variables of the PEP. Indeed, the difference between the performance metric and τ times the initialisation measure of proximity to the optimizer is decomposed as the sum of three terms. The two first ones respectively correspond to the values that are enforced to be negative by the class of functions and the algorithm. The third one is called the residual and is the opposite of a sum of squares of iterates and gradients. An example of full derivation of such a proof is provided in Section V.

Remark 4.1 (No duality gap): There generally exists a feasible point G, F with $G \succ 0$, i.e. verifying the Slater's condition (see [32]), therefore guaranteeing strong duality of the convex reformulation of ($\mathcal{P}$). To ensure this, one needs to carefully remove iterates x_t from the basis of G when x_t is completely identified from other vectors. For instance, leaving x_1 in the basis of G with the constraint $\|x_1 - (x_0 - \gamma g_0)\|^2 = 0$ instead of replacing x_1 by $x_0 - \gamma g_0$ everywhere, creates an empty interior and can break strong duality. Each time there is no feasible G with $G \succ 0$, we conclude that there is a linear relationship between elements of the basis G is the Gram matrix of. Therefore, maximally reducing the dimension of G ensures strong duality.

B. Understanding proofs with PEPs

Obtaining dual feasible points provides valuable insights into essential aspects pertaining to both the class of functions under consideration and the algorithm employed to achieve the associated worst-case guarantee.

Extension to broader sets of algorithms. [13] exploit these insights to *design worst-case optimal algorithms*. The authors' key observation is that (Generic proof) does not rely on all constraints to hold, but rather only on a linear combination of them. Therefore, if instead of assuming that, $\forall l, \left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle \leqslant 0$, we can simply assume that $\sum_l \lambda_{\mathcal{A}}^{(l)}\left[\left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle\right] \leqslant 0$, therefore relaxing a lot of assumptions about the algorithm and then generalizing the proof to all the algorithms verifying the remaining assumption. This was applied to the impractical algorithm (GFOM) described as follow:

$$\forall t, x_{t+1} = \underset{x \in x_0 + \text{span}\{\nabla f(x_0), \cdots, \nabla f(x_t)\}}{\arg\min} f(x), \quad \text{(GFOM)}$$

greedily minimizing the objective value in the affine space of all the observed directions. For some classes of functions, this algorithm is worst-case optimal. This is the case, for instance, for the class of quadratic convex functions on which (GFOM) is equivalent to the so-called *conjugate gradient method*. This is also the case for the class of L-smooth convex functions, allowing to find a broad range of worst-case optimal algorithms on this class, including the so-called *optimized gradient method (OGM)* [12], [13]. Generating such worst-case optimal algorithms works as follow:

1) We note that (GFOM) verifies the following orthogonality constraints:
$$\forall t, \left\{\begin{array}{l} \forall s < t, \langle g_t, g_s\rangle = 0, \\ \forall s \leqslant t, \langle g_t, x_s - x_0\rangle = 0. \end{array}\right. \quad (9)$$
Note that following those constraints does not necessarily imply that (PEP-primal)'s primal variables optimal values describe (GFOM). Nevertheless, a sufficient condition on the class $\mathcal{F}$ under consideration for that to happen is that $\mathcal{F}$ is contraction-preserving (see [13, Definition 3]), which happens to be the case for $\mathcal{F}_{\mu,L}$ for example.

2) We call the corresponding dual variables $(\beta_{t,s})_{s<t}$ and $(\gamma_{t,s})_{s\leqslant t}$ and collect their optimal values $(\beta^\star_{t,s})_{s<t}$ and $(\gamma^\star_{t,s})_{s\leqslant t}$: it happens that those values can be obtained in closed-form.
3) We group all the constraints as in (9), and conclude that the worst-case guarantee of (GFOM), as well as the corresponding proof, would hold if

$$\forall t, \left\langle g_t, \sum_{s=0}^{t-1} \beta_{t,s} g_s + \sum_{s=0}^{t} \gamma_{t,s}(x_s - x_0) \right\rangle \leqslant 0. \quad (10)$$

4) When $\gamma_{t,t} \neq 0$, we conclude that, in particular, the algorithm described by the iteration

$$\forall t, x_t = x_0 - \sum_{s=0}^{t-1} \frac{\gamma_{t,s}}{\gamma_{t,t}}(x_s - x_0) + \frac{\beta_{t,s}}{\gamma_{t,t}} g_s \quad (11)$$

annihilates the vector in the right-hand position of the inner product. Therefore, the worst-case guarantee of (GFOM) also applies to $\mathcal{A}$, using the exact same proof.

This method has more recently been used in [22, Th.2.4-Cor.2.5] to derive the worst-case optimal algorithm

$$x_t = \frac{t}{t+1} x_{t-1} + \frac{1}{t+1} x_0 - \frac{1}{t+1} \sum_{s=0}^{t-1} \frac{1}{L} g_s \quad (\text{HB})$$

under the class of convex and L-quadratically upper bounded (L-QG$^+$) functions.

Extension to broader classes of functions. Interestingly, (HB) was studied several years ago in [18] on the class $\mathcal{F}_{0,L}$ of L-smooth convex functions, itself included in the class of L-QG$^+$ convex functions. On the other hand, the obtained guarantee was not better on $\mathcal{F}_{0,L}$ than the one obtained on the class of L-QG$^+$ convex functions. This shows that the guarantee obtained on $\mathcal{F}_{0,L}$ can be obtained using only the interpolation constraints of the class of L-QG$^+$ convex functions, which is a subset of the set of interpolation constraints of $\mathcal{F}_{0,L}$. In general, for a given class and a given algorithm, when $\lambda^{(k)}_{\mathcal{F}} = 0$ in (Generic proof), we conclude that the corresponding constraint has not been used. This allows to discard all the useless constraints and the result naturally holds on a larger class of functions.

Fewer class constraints allows new algorithms. Most of the time, we study a family of classes of functions, parametrized by some value L. A classical example of this is the class of L-smooth convex functions $\mathcal{F}_{0,L}$. The underlying *interpolation constraints* $\left\langle F, v^{(k)}_{\mathcal{F}}(L) \right\rangle + \left\langle G, M^{(k)}_{\mathcal{F}}(L) \right\rangle \leqslant 0$ then depend on L. We generally derive and study an algorithm on $\mathcal{F}_{0,L}$, and obtain a guarantee that holds for any L such that $\langle F, v_P(L)\rangle + \langle G, M_P(L)\rangle - \tau(L)\left[\langle F, v_I(L)\rangle + \langle G, M_I(L)\rangle\right] \leqslant 0$. The underlying algorithm can (and usually does) therefore depend on this value that is sometimes hard to access in practice. Using line-search steps is a way to get rid of the dependence on L (there exists for instance line-search version of *OGM* and (HB) that do not involve L), but an exact line-search step is often not available neither. On the other hand, *backtracking line-search* have been proposed [1] to replace the class parameter L by any surrogate value $\hat{L}$ that validates all the inequalities that are used. Indeed, we know that for any L,

$$\begin{aligned} &\underbrace{\langle F, v_P(L)\rangle + \langle G, M_P(L)\rangle}_{\text{Performance metric}} - \tau(L) \underbrace{\left[\langle F, v_I(L)\rangle + \langle G, M_I(L)\rangle\right]}_{\text{Initialization}} \\ &\leqslant \sum_j \lambda^{(j)} \underbrace{\left[\left\langle F, v^{(j)}(L)\right\rangle + \left\langle G, M^{(j)}(L)\right\rangle\right]}_{\text{Constraint}} \\ &\leqslant 0 \end{aligned} \quad (12)$$

Therefore, even if we do not have access to L, being able to find some $\hat{L}$ in an online manner such that all the surrogate constraints $\left\langle F, v^{(k)}_{\mathcal{F}}(\hat{L})\right\rangle + \left\langle G, M^{(k)}_{\mathcal{F}}(\hat{L})\right\rangle \leqslant 0$ hold, allows tuning the algorithm online with this $\hat{L}$ and obtain the guarantee $\left\langle F, v_P(\hat{L})\right\rangle + \left\langle G, M_P(\hat{L})\right\rangle - \tau(\hat{L})\left[\left\langle F, v_I(\hat{L})\right\rangle + \left\langle G, M_I(\hat{L})\right\rangle\right] \leqslant 0$. We would like to apply bisection search to find such $\hat{L}$, and all we need for that is being able to verify the constraints $\left\langle F, v^{(k)}_{\mathcal{F}}(\hat{L})\right\rangle + \left\langle G, M^{(k)}_{\mathcal{F}}(\hat{L})\right\rangle \leqslant 0$ online. Note however that some constraints may involve the optimizer $x_\star$ or the minimal value $f_\star$ and are then not verifiable. The authors of [16, Remark 4.9] and [27] discuss this issue. They note that we only need to verify constraint that actually involve L and that the ones that are problematic are the ones that involve both L and an unknown value. They conclude that, if the dual values associated with these problematic constraints are set to 0, they are not used, and then we can proceed to *backtracking line-search*. They also enforce it by removing those inequalities (or lowering them) and searching for methods that holds on this larger class of functions (verifying less inequalities) in order to be able to apply *backtracking line-search* to get rid of the requirement of knowledge of the parameter class.

V. Example: Gradient Descent with Exact Line-Search

For sake of better comprehension of the formal reasoning made in Sections II-B, III and IV, we detail in this section the development of a proof of convergence guarantee of the form (Generic proof) on an example: the gradient descent method with exact line-search, defined as

$$\forall t \in [\![1, T]\!],\ x_t = \operatorname*{arg\,min}_{x \in x_{t-1} + \operatorname{span}\{\nabla f(x_{t-1})\}} f(x). \quad (\text{GDLS})$$

More precisely, we chose to consider the function value as performance metric, and therefore seek for a guarantee of the form

$$f(x_1) - f_\star \leqslant \tau(f_0 - f_\star), \quad (13)$$

with an appropriate τ. Note this problem has been solved in [11, Theorem 1.2]. Here we detail how to find such a guarantee and its proof in a very systematic way, relying on the framework presented in the present tutorial.

The problem can therefore be summarized as follow:

- The objective function belongs to the class $\mathcal{F}_{\mu,L}$ of L-smooth μ-strongly-convex functions, i.e. verifies the interpolation constraints (IC),
- We have access to the oracle $\mathcal{O}^f(x)$ verifying:
  - $\mathcal{O}^f(x) \in x + \text{span}\{\nabla f(x)\}$,
  - $\langle \nabla f\left(\mathcal{O}^f(x)\right), \nabla f(x)\rangle = 0$,
- The algorithm $\mathcal{A}$ we study iteratively computes the update $x_t = A_t((x_s, \mathcal{O}^f(x_s))_{s<t}) \triangleq \mathcal{O}^f(x_{t-1})$,
- We study exactly one step of this algorithm. That is, we want a guarantee on x_1 given x_0.
- The performance metric that we use is the function value $f(x_1) - f_\star$.
- The neighborhood $\mathcal{N}(x_\star)$ we assume x_0 belongs to is also define by the function value as $\left\{x \mid f(x) - f_\star \leqslant R^2\right\}$ for some positive R.

In summary, the problem ($\mathcal{P}$) writes

$$\left|\begin{array}{ll} \underset{\substack{f\in\mathcal{F}_{\mu,L}, d\geqslant 1\\ (x_\star,x_0,x_1)\in\left(\mathbb{R}^d\right)^3}}{\text{maximize}} & f(x_1) - f_\star \\ \text{subject to} & \left\{\begin{array}{l} f(x_0) - f_\star \leqslant R^2 \\ (x_t)_{t\leqslant 1} = \text{GDLS}(f, T=1, x_0) \end{array}\right. \end{array}\right. \tag{14}$$

GDLS's update is defined through an optimization problem. Implementing it into the PEP framework is not straightforward. Instead, we replace the strict definition of the update by first order optimality conditions of the line search procedure:

$$\begin{aligned} \langle \nabla f(x_1), \nabla f(x_0)\rangle &= 0, \\ \langle \nabla f(x_1), x_1 - x_0\rangle &= 0. \end{aligned}$$

Note the second one is verified because $x_1 - x_0$ is colinear with g_0 and therefore those 2 conditions seem redundant. However, removing the proper definition of (GDLS) makes $x_1 - x_0$ and g_0 non-necessarily colinear anymore, and the two orthogonality conditions are complementary.

Note furthermore that, replacing the actual definition of (GDLS) by some conditions the latter verifies leads to a guarantee that holds over all the algorithms that verify those conditions. This is therefore possibly a relaxation, but the result still holds. Moreover, in this special case, and because we used the two orthogonality conditions and not just one, replacing the definition of (GDLS) by those conditions is tight. This technical assertion is based on the fact the class $\mathcal{F}_{\mu,L}$ is *contraction-preserving*. This reasoning is detailed in [13].

Expressing the constraints of the algorithm and the class, we obtain

$$\left|\begin{array}{ll} \underset{\substack{d\geqslant 1, (x_\star,x_0,x_1))\in(\mathbb{R}^d)^3, \\ (g_0,g_1)\in(\mathbb{R}^d)^2,\ (f_\star,f_0,f_1)\in\mathbb{R}^3}}{\text{maximize}} & f(x_1) - f_\star \\ \text{s.t.} & \left\{\begin{array}{l} f(x_0) - f_\star \leqslant R^2 \\ \langle \nabla f(x_1), \nabla f(x_0)\rangle = 0, \\ \langle \nabla f(x_1), x_1 - x_0\rangle = 0. \\ \forall i,j,\ f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i - g_j\|^2 \\ \qquad + \frac{\mu}{2(1-\mu/L)}\|x_i - \frac{1}{L}g_i - x_j + \frac{1}{L}g_j\|^2. \end{array}\right. \end{array}\right.$$

Using SDP lifting, we can formulate this problem as a semi-definite program of the form (PEP-primal) using the variables

$$\begin{aligned} F &= (f_\star, f_0, f_1)^\top \\ G &= (x_\star, x_0, g_0, x_1, g_1)^\top (x_\star, x_0, g_0, x_1, g_1). \end{aligned}$$

We therefore set the parameters of (Generic proof) to the following values:

$$\begin{aligned} v_P &= (-1, 0, 1)^\top, & M_P &= 0_5, \\ v_I &= (-1, 1, 0)^\top, & M_I &= 0_5, \end{aligned}$$

$$v_{\mathcal{F}}^{(\star,0)} = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \quad M_{\mathcal{F}}^{(\star,0)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & -\mu & 1 & 0 & 0 \\ -\mu & \mu & -1 & 0 & 0 \\ 1 & -1 & \frac{1}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$v_{\mathcal{F}}^{(\star,1)} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(\star,1)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & 0 & 0 & -\mu & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\mu & 0 & 0 & \mu & -1 \\ 1 & 0 & 0 & -1 & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{F}}^{(0,\star)} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad M_{\mathcal{F}}^{(0,\star)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & -\mu & \kappa & 0 & 0 \\ -\mu & \mu & -\kappa & 0 & 0 \\ \kappa & -\kappa & \frac{1}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$v_{\mathcal{F}}^{(0,1)} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(0,1)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & -\kappa & -\mu & 1 \\ 0 & -\kappa & \frac{1}{L} & \kappa & -\frac{1}{L} \\ 0 & -\mu & \kappa & \mu & -1 \\ 0 & 1 & -\frac{1}{L} & -1 & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{F}}^{(1,\star)} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(1,\star)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & 0 & 0 & -\mu & \kappa \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\mu & 0 & 0 & \mu & -\kappa \\ \kappa & 0 & 0 & -\kappa & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{F}}^{(1,0)} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(1,0)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & -1 & -\mu & \kappa \\ 0 & -1 & \frac{1}{L} & 1 & -\frac{1}{L} \\ 0 & -\mu & 1 & \mu & -\kappa \\ 0 & \kappa & -\frac{1}{L} & -\kappa & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{A}}^{(1)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad M_{\mathcal{A}}^{(1)} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

$$v_{\mathcal{A}}^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad M_{\mathcal{A}}^{(2)} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 \end{pmatrix}.$$

Solving this SDP, we find the rate $\tau = \left(\frac{L-\mu}{L+\mu}\right)^2$. Moreover, the corresponding dual values are

$$\boxed{\lambda_{\mathcal{F}}^{\star,0} = \frac{2\mu(L-\mu)}{(L+\mu)^2}}, \qquad \boxed{\lambda_{\mathcal{F}}^{\star,1} = \frac{2\mu}{L+\mu}},$$

$$\lambda_{\mathcal{F}}^{0,\star} = 0, \qquad \boxed{\lambda_{\mathcal{F}}^{0,1} = \frac{L-\mu}{L+\mu}},$$

$$\lambda_{\mathcal{F}}^{1,\star} = 0, \qquad \lambda_{\mathcal{F}}^{1,0} = 0,$$

$$\boxed{\lambda_{\mathcal{A}}^{1} = \frac{2}{L+\mu}}, \qquad \boxed{\lambda_{\mathcal{A}}^{2} = 1.}$$

Plugging those values in (Generic proof) builds a proof of convergence of (GDLS) with the guarantee $f(x_1) - f_\star \leqslant \left(\frac{L-\mu}{L+\mu}\right)^2 (f_0 - f_\star)$.

$$
\begin{aligned}
& f(x_1) - f_\star - \left(\frac{L-\mu}{L+\mu}\right)^2 (f(x_0) - f_\star) \\
\leqslant\ & \tfrac{2\mu(L-\mu)}{(L+\mu)^2}\left(f(x_0) - f_\star + \langle \nabla f(x_0), x_\star - x_0\rangle + \tfrac{1}{2L}\|\nabla f(x_0)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\|x_\star - x_0 + \tfrac{1}{L}\nabla f(x_0)\|^2\right) \\
& + \tfrac{2\mu}{L+\mu}\left(f(x_1) - f_\star + \langle \nabla f(x_1), x_\star - x_1\rangle + \tfrac{1}{2L}\|\nabla f(x_1)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\|x_\star - x_1 + \tfrac{1}{L}\nabla f(x_1)\|^2\right) \\
& + \tfrac{L-\mu}{L+\mu}\left(f(x_1) - f(x_0) + \langle \nabla f(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|\nabla f(x_0) - \nabla f(x_1)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\|x_0 - \tfrac{1}{L}\nabla f(x_0) - x_1 + \tfrac{1}{L}\nabla f(x_1)\|^2\right) \\
& + \tfrac{2}{L+\mu}\langle \nabla f(x_1), \nabla f(x_0)\rangle \\
& + \langle \nabla f(x_1), x_1 - x_0\rangle \\
\leqslant\ & 0.
\end{aligned}
$$

The first inequality holds independently on the chosen class. It simply results from terms rearrangement. By subtracting the LHS from the RHS, one would find a semi-definite positive quadratic form of the variables $x_0, x_1, \nabla f(x_0)$ and $\nabla f(x_1)$. The second inequality precisely uses the (in)equalities that are specific to the chosen class and algorithm. Note that the two algorithm constraints can be replaced by the sole constraint $\left\langle \nabla f(x_1), x_1 - x_0 + \frac{2}{L+\mu}\nabla f(x_0)\right\rangle = 0$, immediately showing that this guarantee also holds on the gradient descent method with fixed steps-size $\frac{2}{L+\mu}$.

VI. Lyapunov with PEPs

We saw in Section IV-A that worst-case proofs essentially writes as (Generic proof):

$$
\begin{aligned}
& \underbrace{\langle F, v_P\rangle + \langle G, M_P\rangle}_{\text{Performance metric}} - \tau \underbrace{[\langle F, v_I\rangle + \langle G, M_I\rangle]}_{\text{Initialization}} \\
& \leqslant \sum_j \lambda^{(j)} \underbrace{\left[\left\langle F, v^{(j)}\right\rangle + \left\langle G, M^{(j)}\right\rangle\right]}_{\text{Constraint}} \\
& \leqslant 0. \qquad (15)
\end{aligned}
$$

Namely, the right linear combination of the available constraints upper bounds the difference between the performance metric and τ times the initial value. Sometimes those proofs can be relatively complicated and a simpler one can be desirable. In particular, this is the case when the algorithm under consideration is run for a few iterations. Lyapunov analyses typically allows reducing the worst-case analyses of T iterations to that of a single iteration, and therefore reducing the complexity of the proof.

For example, for (NAG), described as follow

$$
\begin{aligned}
\lambda_{t+1} &= \tfrac{1}{2} + \sqrt{\tfrac{1}{4} + \lambda_t^2} \\
y_t &= x_t + \tfrac{\lambda_t - 1}{\lambda_{t+1}}(x_t - x_{t-1}), \\
x_{t+1} &= y_t - \tfrac{1}{L}\nabla f(y_t).
\end{aligned} \qquad \text{(NAG)}
$$

on $\mathcal{F}_{0,L}$, we often use the sequence

$$
V_t = \lambda_t^2(f_t - f_\star) + \frac{L}{2}\|\lambda_t(x_t - x_\star) + (1-\lambda_t)(x_{t-1} - x_\star)\|^2 \qquad (16)
$$

providing a worst-case convergence guarantee $f(x_T) - f_\star = \mathcal{O}(1/T^2)$. In general, a direct way to find such a sequence is to consider

$$
\begin{aligned}
V_t = & \underbrace{[\langle F, v_I\rangle + \langle G, M_I\rangle]}_{\text{Initialization}} \\
& + \sum_{j \mid \substack{\text{only involves} \\ \text{values observed} \\ \text{before step } t}} \lambda^{(j)} \underbrace{\left[\left\langle F, v^{(j)}\right\rangle + \left\langle G, M^{(j)}\right\rangle\right]}_{\text{Constraint}}. \qquad (17)
\end{aligned}
$$

Applying this method on (NAG) provides the sequence of complete potential functions

$$
\begin{aligned}
V_t = & \lambda_t^2(f_t - f_\star) + \frac{L}{2}\|\lambda_t(x_t - x_\star) + (1-\lambda_t)(x_{t-1} - x_\star)\|^2 \\
& + \frac{1}{2L}\sum_{s=1}^{t-1}[\lambda_{s+1}^2\|\nabla f(x_{s+1})\|^2 + \lambda_{s+1}\|\nabla f(y_s)\| \\
& \qquad + \lambda_s^2\|\nabla f(y_s) - \nabla f(x_s)\|^2]
\end{aligned}
$$

that allows for free (using the same inequalities as for proving that (16) is decreasing) to also conclude that $\min_{t\leqslant T}\|\nabla f(x_t)\|^2 = \mathcal{O}(1/T^3)$, as shown in [25, Theorem 5.2.d] and experimentally evidenced using PEPs in [36, Table 4].

Note that the cumulatively summed up constraints involve both class constraints and algorithm constraints. Therefore, this technique can be applied directly on (GFOM) while looking for an optimal algorithm, its rate, the corresponding proof and a sequence of potential functions at the same time.

VII. Conclusion

a) Summary: not only is the *performance estimation problem (PEP)* framework a powerful tool to automate the search of guarantees, but also it allows exhibiting general structure of proofs. Understanding this structure enables to generalize results onto larger class of functions or onto a class of methods, but also to find new optimization methods and study their convergence properties. Finally, it also enables to understand how to build a Lyapunov sequence of functions.

b) Open research directions: all this framework relies on two major assumptions: the class constraints are known and homogeneous in $\|x\|^2$ and $\|\nabla f(x)\|^2$ and f, and the method's update is a linear combination of previous iterates and observed oracle calls. Therefore, two interesting questions arise: can we automate the search of the interpolation conditions? And, how can we generalize this framework to non homogeneous class of functions or to non linear methods such as adaptive step-size based methods? A few works already investigate this direction for some specific methods. In particular, [3] studies a variant of the Heavy-ball method [29] using Polyak step-sizes, also discussed in [2, Chapter 4]. On the other hand, [24] uses PEP techniques to provide worst-case guarantees on several variants of non-linear conjugate gradient methods.

ACKNOWLEDGMENTS

The work of B. Goujaud and A. Dieuleveut is partially supported by ANR-19-CHIA-0002-01/chaire SCAI, and Hi!Paris. A. Taylor acknowledges support from the European Research Council (grant SEQUOIA 724063). This work was partly funded by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

REFERENCES

[1] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 1966.

[2] M. Barré. *Worst-case analysis of efficient first-order methods*. PhD thesis, Université Paris sciences et lettres, 2021.

[3] M. Barré, A. Taylor, and A. d'Aspremont. Complexity guarantees for polyak steps with momentum. In *Conference on Learning Theory*, pages 452–478. PMLR, 2020.

[4] R. Berthier, F. Bach, and P. Gaillard. Accelerated gossip in networks of given dimension using jacobi polynomial iterations. *SIAM Journal on Mathematics of Data Science*, 2020.

[5] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007.

[6] S. Bubeck. Convex optimization: Algorithms and complexity. *Found. and Trends in Machine Learning*, 2015.

[7] A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 1847.

[8] A. Chambolle and T. Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 2016.

[9] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. *Fixed-point algorithms for inverse problems in science and engineering*, 2011.

[10] L. Cunha, G. Gidel, F. Pedregosa, D. Scieur, and C. Paquette. Only tails matter: Average-case universality and robustness in the convex regime. In *ICML*, 2022.

[11] E. De Klerk, F. Glineur, and A. B. Taylor. On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions. *Optimization Letters*, 2017.

[12] Y. Drori. The exact information-based complexity of smooth convex minimization. *Journal of Complexity*, 39:1–16, 2017.

[13] Y. Drori and A. B. Taylor. Efficient first-order methods for convex minimization: a constructive approach. *Math. Progr.*, 2020.

[14] Y. Drori and M. Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Math. Progr.*, 2014.

[15] P. Dvurechensky, S. Shtern, and M. Staudigl. First-order methods for convex optimization. *EURO J. on Computational Optimization*, 2021.

[16] A. d'Aspremont, D. Scieur, and A. Taylor. Acceleration methods. *Found. and Trends in Optimization*, 2021.

[17] B. Fischer. *Polynomial based iteration methods for symmetric linear systems*. SIAM, 2011.

[18] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson. Global convergence of the Heavy-ball method for convex optimization. In *ECC*, 2015.

[19] G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods. *Numerische Mathematik*, 1961.

[20] B. Goujaud, C. Moucer, F. Glineur, J. Hendrickx, A. Taylor, and A. Dieuleveut. PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python. *arXiv:2201.04040*, 2022.

[21] B. Goujaud, D. Scieur, A. Dieuleveut, A. B. Taylor, and F. Pedregosa. Super-acceleration with cyclical step-sizes. In *AISTATS*, 2022.

[22] B. Goujaud, A. Taylor, and A. Dieuleveut. Optimal first-order methods for convex functions with a quadratic upper bound. *arXiv:2205.15033*, 2022.

[23] B. Goujaud, A. Taylor, and A. Dieuleveut. Quadratic minimization: from conjugate gradient to an adaptive Heavy-ball method with Polyak step-sizes. *arXiv:2210.06367*, 2022.

[24] S. D. Gupta, R. M. Freund, X. A. Sun, and A. Taylor. Nonlinear conjugate gradient methods: worst-case convergence rates via computer-assisted analyses. *arXiv preprint arXiv:2301.01530*, 2023.

[25] R. D. Monteiro and B. F. Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.

[26] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 1983.

[27] C. Park and E. K. Ryu. Optimal first-order algorithms as a function of inequalities. *arXiv:2110.11035*, 2021.

[28] F. Pedregosa and D. Scieur. Acceleration through spectral density estimation. In *ICML*, 2020.

[29] B. T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 1963.

[30] D. Scieur. *Acceleration in optimization*. PhD thesis, 2018.

[31] D. Scieur and F. Pedregosa. Universal average-case optimality of Polyak momentum. In *ICML*, 2020.

[32] M. Slater. Lagrange multipliers revisited: a contribution to nonlinear programming, 1950.

[33] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[34] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Exact worst-case performance of first-order methods for composite convex optimization. *SIAM J. on Optimization*, 2017.

[35] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Performance estimation toolbox (PESTO): automated worst-case analysis of first-order optimization methods. In *CDC*, 2017.

[36] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Math. Progr.*, 2017.