Performance Estimation Toolbox (PESTO): automated worst-case analysis of first-order optimization methods

Adrien B. Taylor, Julien M. Hendrickx, François Glineur

Abstract— **We present a MATLAB toolbox that automatically computes tight worst-case performance guarantees for a broad class of first-order methods for convex optimization. The class of methods includes those performing explicit, projected, proximal, conditional and inexact (sub)gradient steps.**

The toolbox relies on the *performance estimation* (PE) framework, which recently emerged through works of Drori and Teboulle and the authors. The PE approach is a very systematic manner of obtaining non-improvable worst-case guarantees for first-order numerical optimization schemes. However, using the PE methodology requires modelling efforts from the user, along with some knowledge of semidefinite programming. The goal of this work is to ease the use of the performance estimation methodology, by providing a toolbox that implicitly does the modelling job. In short, its aim is to (i) let the user write the algorithm in a natural way, as he/she would have implemented it, and (ii) let the computer perform the modelling and worst-case analysis parts automatically.

CODE

The toolbox is fully available on GITHUB (repository ADRIENTAYLOR/PERFORMANCE-ESTIMATION-TOOLBOX). The code heavily relies on the YALMIP modelling language [1] and on the use of an appropriate SDP solver (e.g., [2], [3], [4]).

The authors would like to particularly thank Yoel Drori (Google Inc.) and François Gonze (UCLouvain) for insightful feedbacks on preliminary versions of the toolbox.

I. INTRODUCTION

Worst-case analysis belongs to the most iconic and widespread generic approaches for assessing the efficiency of numerical algorithms (e.g., worst-case iteration complexity). Essentially, given a class of problems and an algorithm designed to solve them, worst-case analysis focuses on the problem instances on which the algorithm behaves in the worst possible way — the meaning of *worst* intentionally remains vague at this point. The user may then be satisfied by the algorithm if its behavior on the worst possible problem instance meets some quality criterion, such as the last iterate being close enough to an optimal point.

This work is supported by the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, and of the Concerted Research Action (ARC) programme supported by the Federation Wallonia-Brussels (contract ARC 14/19-060). The scientific responsibility rests with its authors.

A.B. Taylor and J.M. Hendrickx are members of ICTEAM, Université catholique de Louvain, Belgium. adrien.taylor, julien.hendrickx@uclouvain.be.

F. Glineur is a member of CORE and ICTEAM, Université catholique de Louvain, Belgium. francois.glineur@uclouvain.be.

The purpose of this work is to provide easy access to the performance estimation methodology and to automatically perform the worst-case analyses of first-order optimization schemes. This allows researchers to very quickly assess the performance of their optimization methods, easing the development of new algorithms.

Before getting to the point, note that multiple alternatives to worst-case analyses exist, such as average-case or smoothed analyses [5]. However, to the best of our knowledge, those approaches have not yet been applied to first-order methods for convex optimization.

A. Problem settings and formulation

In the sequel, we focus on algorithms tailored for solving convex composite minimization (CCM) problems:

$$\min_{x\in\mathbb{R}^d} \left\{ F(x) = \sum_{k\in K} F^{(k)}(x) \right\}, \tag{CCM}$$

where each component $F^{(k)} : \mathbb{R}^d \to \mathbb{R}$ is assumed to belong to some class of proper, closed and convex functions $\mathcal{F}_k(\mathbb{R}^d)$, and K is a finite index set. As an example, a common particular case of (CCM) is the minimization of a single component objective function (i.e., $|K| = 1$).

In what follows, we focus on iterative algorithms using only first-order information (i.e., gradients and subgradients) in order to solve (CCM). In particular, we are interested in the worst-case performances of those methods, which we approach by computing the functions $F(x)$ on which the methods behave in the worst possible way.

References on the topic of using first-order methods for solving (CCM) include the works of Nemirovski and Yudin [6], Polyak [7] and Nesterov [8].

B. Related works

Performance estimation problems were initially developed in the seminal work of Drori and Teboulle [9]. In their work, they formulate the problem of finding the worst function $F(x)$ in (CCM) for fixed-step first-order methods. Their formulation allows among others treating the gradient, the heavy-ball and the fast gradient methods when F consists of a single smooth convex component, that is, $|K| = 1$ and $\mathcal{F}_1(\mathbb{R}^d) = \mathcal{F}_{0,L}(\mathbb{R}^d)$, the class of L-smooth convex functions (defined in the sequel).

In order to obtain guarantees on the worst-case behaviors of those methods, they discretize, relax and dualize the worst-case computation problem, and finally end up with a convex semidefinite program (SDP) whose feasible solutions are worst-case guarantees. They provide multiple numerical and analytical examples showing that their approach improves the standard guarantees on fixed-step methods, see also Drori's PhD thesis [10] for more examples.

The treatment of the worst-case computation problem is made more systematic in the works of the authors [11], [12], which add a key element to the equation: a proper non-conservative discretization scheme relying on the *convex interpolation* framework. This allows to generically formulate the worst-case computation problem in a way that ensures the guarantees to be tight, that is, such that there always exist functions on which those worst-case guarantees are achieved, making them non-improvable in general. Also, the approach is extended to provide tight worst-case guarantees for a larger class of first-order methods, including projected, proximal, conditional and inexact gradient schemes, see also [13], [14] for additional examples.

On the application side, this new methodology was used to design new optimization schemes. Starting from the original work of Drori and Teboulle [9], the technique is used to numerically design a new method, the optimized gradient method (OGM), whose parameters are chosen by optimizing the worst-case guarantee provided by their SDP relaxations. This method is further studied in the work of Kim and Fessler [15], who found an efficient analytical form for it. In addition, they show that OGM has a worst-case guarantee twice better than the celebrated fast gradient method developed by Nesterov [16], at essentially the same computational cost per iteration. Recently, Drori showed that OGM achieves the best possible worst-case guarantee for smooth convex minimization in [17], and OGM and its variants were further studied by Kim and Fessler in [18], [19]. Drori and Teboulle also used related ideas to develop a new bundle-like method achieving the best possible worst-case iteration complexity for non-smooth convex minimization in [20].

Finally, the approach was used for developing new (tight) guarantees for the steepest gradient descent in [13], and was largely extended in [14] to handle a larger class of algorithms, including splitting and randomized methods.

Another closely related technique was developed by Lessard, Recht and Packard in [21]. It relies on interpreting fixed-step first-order methods applied to convex optimization problems as dynamical systems, whose convergence can be studied using the related stability theory. This methodology is specialized to study *time-invariant* methods with linear convergence rates. It does not benefit a priori from tightness guarantees, but leads to asymptotic convergence rates. However, the method has the advantage of only requiring the resolution of small semidefinite programs, whereas the PE approach may require solving much larger SDPs, whose dimensions scale proportionally to the number of iterations.

C. Organization

The remainder of this work is divided into four main sections. In Section II, we present the framework on a simple toy example before introducing the general performance estimation approach. Then, Section III introduces the toolbox structure and content. We provide two applicative examples in Section IV, before drawing some conclusions in Section V.

II. PERFORMANCE ESTIMATION PROBLEMS

In this section, we briefly introduce the performance estimation approach using the philosophy and formalism from [11] and [12]. We motivate the general approach starting from a simple example: obtaining the worst-case guarantees of a single gradient step on a smooth strongly convex function. Before doing that, let us recall some definitions and notations. First, a differentiable function $f:\mathbb{R}^d\to\mathbb{R}$ is

- μ-strongly convex if and only if $f(x)-\frac{\mu}{2}\|x\|^2$ is convex,
- L-smooth if and only if

$$\|\nabla f(x)-\nabla f(y)\|\leq L\|x-y\| \quad \forall x,y\in\mathbb{R}^d.$$

We denote by $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ the set of L-smooth μ-strongly convex functions, and more generally by $\mathcal{F}_{0,\infty}(\mathbb{R}^d)$ the class of convex, closed and proper functions $f:\mathbb{R}^d\to\mathbb{R}\cup\{+\infty\}$ (i.e., functions whose epigraphs are convex, closed and non-empty). Finally, we denote by $\partial f(x)$ the subdifferential of f at x, that is:

$$\partial f(x)=\{g_x:\ f(y)\geq f(x)+\langle g_x,y-x\rangle\ \forall y\in\mathbb{R}^d\},$$

and by x_* an optimal point of (CCM).

A. Toy example: worst-case analysis of a gradient step

In this section, we exemplify the performance estimation approach by answering the following simple question.

Let $F\in\mathcal{F}_{\mu,L}(\mathbb{R}^d)$; what can we guarantee on $\|\nabla F(x_1)\|$ when x_1 was obtained from a gradient step $x_1=x_0-\gamma\nabla F(x_0)$ with step size γ ?

For this question to be well posed, we introduce an initial condition, as otherwise the worst-case value $\|\nabla F(x_1)\|$ would be unbounded. Here, we arbitrarily choose to assume

$$\|\nabla F(x_0)\|^2\leq R^2 \quad (R \text{ is some constant parameter}).$$

Then, we want to answer the previous question without being conservative (i.e., we want the *exact*, or *tight* guarantee), and therefore choose to formulate it as the optimal value to the following worst-case computation problem, or performance estimation problem (PEP):

$$\begin{aligned} \max_{F,x_0,x_1}\ & \|\nabla F(x_1)\|^2 && \text{(Gradient-PEP)}\\ \text{s.t. } & F\in\mathcal{F}_{\mu,L}(\mathbb{R}^d) && \text{Functional class,}\\ & x_1=x_0-\gamma\nabla F(x_0) && \text{Algorithm,}\\ & \|\nabla F(x_0)\|^2\leq R^2 && \text{Initial condition.}\end{aligned}$$

Observe that this problem is inherently infinite-dimensional, as it contains the function F among the variables (this is essential, as the goal is to identify the worst-case function). To deal with this variable, we first note that we only evaluate the values and gradients of the function at certain points: the iterates. This motivates the introduction of new variables: the iterates x_0,x_1, and the corresponding gradients g_0,g_1, and function values f_0,f_1. We then require the existence of a function of the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ which *explains* (or interpolates) those data points, instead of considering the whole function F:

$$\max_{\{(x_i,g_i,f_i)\}_{i=0,1}} \|g_1\|^2 \qquad \text{(Discr-Gradient-PEP)}$$
$$\text{s.t. } \exists F \in \mathcal{F}_{\mu,L}(\mathbb{R}^d) : f_i = F(x_i),\ g_i = \nabla F(x_i)\ (i=0,1),$$
$$x_1 = x_0 - \gamma g_0,$$
$$\|g_0\|^2 \leq R^2.$$

Thanks to the first constraint guaranteeing the existence of an interpolating function $F \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, the optimal values of (Discr-Gradient-PEP) and (Gradient-PEP) are equal. Indeed, any feasible solution to one of those problems can be converted to a feasible solution for the other one.

In order to transform this last formulation into a tractable problem, we use the following *smooth strongly convex interpolation theorem.*

Theorem 1: [12, Theorem 4] Let I be an index set and $S = \{(x_i, g_i, f_i)\}_{i\in I}$ be such that $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. There exists a function $F \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ such that $f_i = F(x_i)$ and $g_i = \nabla F(x_i)$ $(i \in I)$ if and only if for all pairs $i \neq j \in I$ we have

$$\begin{aligned} f_i \geq\ & f_j + \langle g_j, x_i - x_j\rangle + \tfrac{1}{2(1-\mu/L)}\left(\tfrac{1}{L}\|g_i - g_j\|^2 \right. \\ & \left. + \mu\|x_i - x_j\|^2 - 2\tfrac{\mu}{L}\langle g_j - g_i, x_j - x_i\rangle\right). \end{aligned} \tag{1}$$

From the smooth strongly convex interpolation theorem, one can reformulate (Discr-Gradient-PEP) in the following equivalent form:

$$\max_{\{(x_i,g_i,f_i)\}_{i=1,2}} \|g_1\|^2 \qquad \text{(QCQP-Gradient-PEP)}$$
$$\text{s.t. interpolation constraints (1) hold } \forall i \neq j \in \{0,1\},$$
$$x_1 = x_0 - \gamma g_0,$$
$$\|g_0\|^2 \leq R^2.$$

This new formulation (QCQP-Gradient-PEP) is a non-convex quadratically constrained quadratic program, a class of problems that is NP-hard in general. To tackle the problem, we perform the following simple operations:

- we substitute x_1 from the variables using the gradient step constraint $x_1 = x_0 - \gamma g_0$,
- we introduce a 3×3 Gram matrix:

$$G = \begin{pmatrix} \langle x_0, x_0\rangle & \langle g_0, x_0\rangle & \langle g_1, x_0\rangle \\ \langle x_0, g_0\rangle & \langle g_0, g_0\rangle & \langle g_1, g_0\rangle \\ \langle x_0, g_1\rangle & \langle g_0, g_1\rangle & \langle g_1, g_1\rangle \end{pmatrix} \succeq 0, \tag{2}$$

  with the following known facts:
  - given any set of vectors $x_0, g_0, g_1 \in \mathbb{R}^d$, we have $G \succeq 0$, and G has rank at most d,
  - given a matrix $G \succeq 0$ and $\operatorname{rank} G \leq d$, we can recover $x_0, g_0, g_1 \in \mathbb{R}^d$ such that G is given by (2).

This allows reformulating (QCQP-Gradient-PEP) into an equivalent rank-constrained semidefinite program (SDP):

$$\max_{G\in\mathbb{S}^3, f\in\mathbb{R}^2} G_{3,3} \qquad \text{(SDP-Gradient-PEP(d))}$$
$$\begin{aligned} \text{such that } f_j - f_i + \operatorname{Tr}(GA_{ij}) &\leq 0, \qquad i \neq j \in \{0,1\}, \\ G_{2,2} - R^2 &\leq 0, \\ G &\succeq 0, \\ \operatorname{rank} G &\leq d. \end{aligned}$$

(where constant matrices A_{01} and A_{10} are obtained from (1), see [12] Section 3.3). Note that the optimal value of (SDP-Gradient-PEP(d)) is an increasing function of the dimension d, and that the rank constraint becomes void as soon as $d \geq 3$. Hence, solving the following convex relaxation allows obtaining worst-case guarantees independently of the value of d:

$$\max_{G\in\mathbb{S}^3, f\in\mathbb{R}^2} G_{3,3} \qquad \text{(SDP-Gradient-PEP)}$$
$$\begin{aligned} \text{such that } f_j - f_i + \operatorname{Tr}(GA_{ij}) &\leq 0, \qquad i \neq j \in \{0,1\}, \\ G_{2,2} - R^2 &\leq 0, \\ G &\succeq 0. \end{aligned}$$

In addition, we have the guarantee that the optimal value of (SDP-Gradient-PEP) can be achieved by a worst-case function of dimension d at most 3. Furthermore, this optimal value is the best possible guarantee with no dependence on d, and it is valid for any value of d (this is called the large-scale setting in standard references, see for example [8]).

The advantage of formulation (SDP-Gradient-PEP) as compared to the previous ones is our ability to solve it numerically to global optimality.

In the case at hand, i.e. that of one step of the gradient method for minimizing a smooth strongly convex function, it turns out it is also feasible to find the optimal value analytically (see [22, Theorem 3.2]), which is given by

$$\max\{\|\nabla F(x_1)\|^2\} = \max\{(1-\gamma\mu)^2, (1-\gamma L)^2\}R^2,$$

with the functions achieving those two worst-case values being respectively the one-dimensional functions $F(x) = \frac{\mu}{2}|x|^2$ and $F(x) = \frac{L}{2}|x|^2$. Hence, the relaxation of the rank constraint had no effect in this particular case: this is in fact common for this kind of methods (see [12, Section 4]).

B. Performance estimation problems: general case

Let us now describe the current approach for handling other functional classes, methods, performance measures and initial conditions (see [11], [14] for the details). We consider the composite minimization model (CCM), a performance measure $\mathcal{P}$ such as $\|\nabla F(x_N)\|^2$ or $\|x_N - x_*\|^2$, and a first-order method $\mathcal{M}$. Using the index set $I = \{*, 0, 1, \ldots, N\}$, the general performance estimation problem of interest is

$$\sup_{\{F^{(k)}\}_{k\in K},\{x_i\}_{i\in I}} \mathcal{P}\left(\left\{x_i, \nabla F^{(k)}(x_i), F^{(k)}(x_i)\right\}_{i\in I, k\in K}\right) \tag{PEP}$$

$$\begin{aligned}\text{s.t. } & F^{(k)} \in \mathcal{F}_k(\mathbb{R}^d) \quad \text{for all } k \in K,\\ & x_0 \text{ satisfies some initialization conditions,}\\ & x_i \text{ is computed by } \mathcal{M} \text{ for all } 1 \le i \le N,\\ & x_* \text{ is a minimizer of } F(x).\end{aligned}$$

Essentially, (PEP) can be translated into a linear SDP similar to (SDP-Gradient-PEP(d)), with a corresponding convex relaxation similar to (SDP-Gradient-PEP)), as soon as the method, interpolation conditions, performance measure and initial conditions can be formulated linearly (and finitely) in terms of the function values and of the entries of a Gram matrix containing scalar products between iterates and (sub)gradients. In that case, all those ingredients are called *linearly Gram-representable* [11, Section 2.2].

The *linearly Gram-representability* assumption is met in surprisingly many standard situations, including:

- *functions*: the class of smooth (possibly strongly) convex functions, of smooth and non-smooth convex functions involving bounded (sub)gradients, of (possibly strongly) convex functions involving bounded domains, of convex indicator and support functions, or even the class of non-convex smooth functions;
- *methods*: first-order methods involving (sub)gradient, projection, proximal, conditional or even inexact steps;
- *performance measure*: $F(x_N) - F(x_*)$, $\|x_N - x_*\|^2$, $\|\nabla F(x_N)\|^2$, $\min_{0\le i\le N} F(x_i) - F(x_*)$, etc.;
- *initialization*: $F(x_0) - F(x_*) \le R$, $\|x_0 - x_*\|^2 \le R^2$, $\|\nabla F(x_0)\|^2 \le R^2$, etc.

All those ingredients are detailed in [11, Section 2]; the corresponding interpolation procedures (i.e., how to construct the interpolating functions in practice) are provided in [14, Chapter 3].

III. PERFORMANCE ESTIMATION TOOLBOX

From the previous sections, the modelling process coming along with the performance estimation approach may appear as a difficult or even a discouraging task. This observation motivates the introduction of PESTO, a MATLAB toolbox .

The main goal is to alleviate the modelling process as much as possible, by providing a framework in which the user can describe the algorithm nearly as he would have implemented it. PESTO then takes care of the worst-case analysis, relying on the PE approach.

The following paragraphs illustrate the use of the toolbox to perform a worst-case analysis for N steps of the gradient method (the setting slightly differs from Section II).

A. Short example: the gradient method

First of all, the main elements are manipulated/initialized by a `pep` object. Such an object is initialized as follows.

```
% Initialize an empty PEP:
P=pep();
```

1) Setting up the objective function: The following step is to specify the family of objective function (CCM) that is to be minimized by the algorithm. For the gradient example, it suffices to write the following lines.

```
param.mu=0.1; % Strong convexity parameter
param.L=1;    % Smoothness parameter

% F is the objective function
F=P.DeclareFunction('SmoothStronglyConvex',param);
```

2) Setting up the initial conditions: Introducing a starting point x_0 can be done via the following line (this can be repeated to generate multiple starting points).

```
% x0 is some starting point:
x0=P.StartingPoint();
```

The next step is then to specify an initial condition: we choose to bound the initial objective function accuracy $F(x_0) - F(x_*) \le 1$.

```
% xs is an optimal point, and fs=F(xs):
[xs,fs]=F.OptimalPoint();

% Initial condition F(x0)-F(xs)<=1:
P.InitialCondition(F.value(x0)-fs<=1);
```

3) Description of the algorithm: We perform the worst-case analysis for N iterations of the gradient method with fixed step sizes γ_i (we define and vary the value of the parameters γ_i and N in the numerical examples from Section IV).

```
x=x0;
for i=1:N
    % Overwrite the previous value of x:
    x=x-gamma(i)*F.gradient(x);
end
xN=x;
```

4) Performance criterion: Finally, we pick the performance criterion to be the residual gradient norm $\|\nabla F(x_N)\|^2$.

```
% Evaluate the gradient at the last iterate:
gN=F.gradient(xN);

% Worst-case evaluated as ||gN||^2:
P.PerformanceMetric(gN^2);
```

5) Worst-case analysis: The performance estimation problem is then solved by typing `P.solve()`. The results can be found in the numerical examples from Section IV.

6) Output: After solving the PEP, any iterate, gradient or function value used and saved in the modelling process can be evaluated using the `double()` command. For example, `double(gN^2)` would output the value of the performance measure, and `double(gN)` would output a valid vector g_N.

B. *Toolbox content*

The toolbox structure is primarily designed to favor the simplicity of the user's code. It is easy to add new functional classes, primitive operations (e.g., gradient steps, projections, etc.) and new types or oracles (e.g., subgradients, inexact subgradients). More details and further examples can be found in the toolbox demonstration files available on GITHUB.

1) Functional classes: The functional classes available within the current version of the toolbox are the following:

- convex functions,
- smooth strongly convex functions,
- smooth convex functions with bounded subdifferentials,
- strongly convex functions with bounded domains,
- indicator functions with bounded domains,
- support functions with bounded subdifferentials,
- non-convex smooth functions.

2) Basic oracles: It is currently possible to evaluate two types of first-order oracles within the toolbox:

- (sub)gradients,
- inexact (sub)gradients (absolute [11] and relative [13] inaccuracy).

3) Basic algorithmic operations: The following basic algorithmic operations are currently available:

- gradient steps,
- projection/proximal steps,
- linear optimization steps (e.g., for conditional gradient),
- exact line-search steps.

4) Performance measures: The default performance criterion is set to be the minimum among all specified performance metrics $\min_i\{P_i(f,G)\}$, where each performance metric $P_i(f,G)$ is an affine combination of the function values and of the scalar products between gradients and coordinates (entries of the Gram matrix). This allows dealing with performance measures such as $\min_{0\leq i\leq N}\{F(x_i)-F(x_*)\}$.

5) Initial conditions: Again we allow to bound any affine combination of the function values and of the scalar products between gradients and coordinates.

Finally, the toolbox contains an extensive list of examples applied to many algorithms (including subgradient, conditional gradient and splitting methods), as well as an exploration of properties of convex functions not directly related to optimization algorithms, along with step-by-step demo files to familiarize users with the code. Typing `demo` in the prompt provides a quick summary of what is available.

IV. NUMERICAL EXAMPLES

The code from the following numerical examples is available within the toolbox. Those examples illustrate the use of the toolbox for settings for which no comparable results were found in the literature. However, detailed comparisons with the standard results on first-order methods can be found in the main references on PEPs by Drori and Teboulle [9], Drori [10] and the authors [11], [12], [14].

A. *Gradient method*

In this section, we execute the code proposed in Section III for evaluating the worst-case value of $\|\nabla F(x_N)\|^2$ for the gradient method when the initial objective function accuracy satisfies $F(x_0)-F(x_*)\leq 1$. We perform the analysis for various step sizes γ in the interval $[0,\frac{2}{L}]$, and for $N\in\{2,5,10\}$; the results are shown on Figure 1.

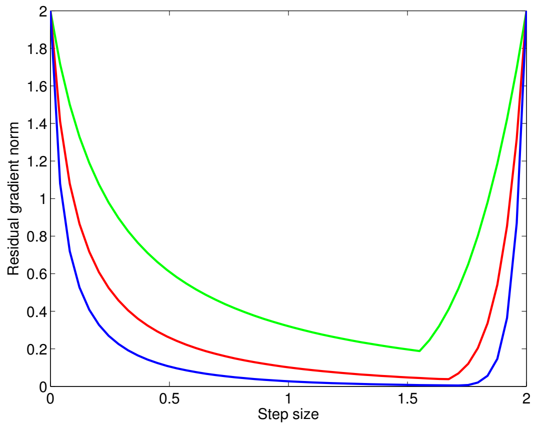

Fig. 1. Gradient method for minimizing a 1-smooth 0.1-strongly convex function. Worst-case value of $\|\nabla f(x_N)\|^2$ as a function of the step size $\gamma\in[0,\frac{2}{L}]$ for $N=2$ (green), $N=5$ (red), and $N=10$ (blue).

From Figure 1, one can observe that this kind of methodology can serve to (i) determine worst-case guarantees for the method (even if this was previously known in this simple case), and (ii) determine optimal step size policies for the settings under consideration; for example, it appears that using the classical so-called *optimal* step size $1/L$ is not the best choice for minimizing the worst-case. Indeed, for all three examples, the optimal step size is an increasing function of N lying within $[\frac{1.5}{L},\frac{2}{L}]$. The choice of the optimal step size very roughly allows to gain a factor 2 on the worst-case behavior. Also, evaluating the worst-cases allows to (iii) observe the shapes of the identified worst-case functions, which can be used to develop new insights on the methods. In this case, the worst-case functions are one-dimensional Huber losses and quadratics.

B. *A fast gradient method using inexact information*

In this section, we report a worst-case analysis on the fast gradient method for minimizing an L-smooth convex function F using inexact gradients $\tilde{\nabla}F$ satisfying a relative accuracy criterion (i.e., bounded relative error):

$$\|\tilde{\nabla}F(x_i)-\nabla F(x_i)\|\leq\varepsilon\|\nabla F(x_i)\|. \tag{3}$$

```
Fast Gradient Method (FGM)
    Input: $F\in\mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0=y_0\in\mathbb{R}^d$.
    For $i=1:N$
        $y_i=x_{i-1}-\frac{1}{L}\nabla F(x_{i-1})$
        $x_i=y_i+\frac{i-1}{i+2}(y_i-y_{i-1})$
```

The PE methodology easily allows studying FGM using this inexact first-order information, replacing the gradient evaluation by its approximate version $\tilde{\nabla} F(x_i)$. In this example, we assume an initial condition $\|x_0 - x_*\| \leq 1$, and evaluate the objective function accuracy $F(x_N) - F(x_*)$ in the worst-case. Numerical results for $N \in \{1, ..., 30\}$ and $\varepsilon \in \{0, 0.1, 0.3, 0.5\}$ are reported on Figure 2.

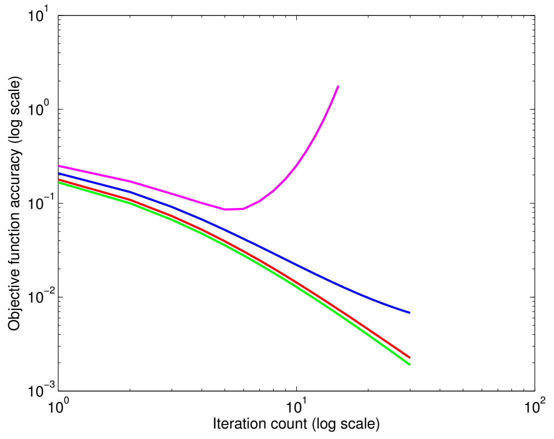

Fig. 2. Fast gradient method for minimizing a 1-smooth convex function using an inexact gradient with bounded relative error. Worst-case value of $F(x_N) - F(x_*)$ as a function of the iteration counter for $\varepsilon = 0$ (green), $\varepsilon = 0.1$ (red), $\varepsilon = 0.3$ (blue), and $\varepsilon = 0.5$ (magenta).

Although the model is, as far as we know, not used in the literature, we arrive to similar conclusions as in [23]: the fast gradient method accumulates error, and can apparently only reach a certain minimum accuracy before starting to diverge, as we suspect from the case $\varepsilon = 50\%$.

We can also observe that after $N = 30$ iterations, a relative accuracy of $\varepsilon = 30\%$ degrades the worst-case objective function accuracy by about 250% compared to the exact case, whereas using a relative accuracy of $\varepsilon = 10\%$ only degrades the worst-case accuracy by about 20%. In others words, $N = 30$ iterations of FGM with $\varepsilon = 10\%$ allows obtaining the same guarantee as $N = 27$ with $\varepsilon = 0$.

V. Conclusion

In this work, we introduced PESTO, a performance estimation toolbox whose goal is to give an easy access to the performance estimation methodology, alleviating the need for demanding modelling steps. We have demonstrated the use of PESTO on a gradient and a fast gradient method, showing that the worst-case analyses could be performed using only a few lines of code, whereas the equivalent full developments could potentially represent a theoretical challenge.

The main design choice of the toolbox, focus on the simplicity of the formulations, comes at the cost of reduced computational efficiency — essentially caused by the preprocessing steps. Advanced users can however easily design their own (more efficient) codes for solving PEPs, tailored for their particular applications. PESTO can then be used for validating those new codes.

The approach allows obtaining numerical bounds on the worst-case performances of the algorithms under consideration. Obtaining analytical guarantees is, on the other hand, often more complicated, and typically requires to perform the modelling process that was avoided by the use of the toolbox. Indeed, developing analytical guarantees coming along with the PE methodology usually amounts to finding feasible solutions to the dual SDP problems (examples in [9], [10], [11], [13]), which is performed on a case-by-case basis.

References

[1] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, 2004.

[2] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, "Sdpt3a matlab software package for semidefinite programming, version 1.3," *Optimization methods and software*, vol. 11, no. 1-4, pp. 545–581, 1999.

[3] A. Mosek, "The MOSEK optimization software," *Online at http://www.mosek.com*, vol. 54, 2010.

[4] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11–12, pp. 625–653, 1999.

[5] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 385–463, 2004.

[6] A. S. Nemirovski and D. B. Yudin, "Problem complexity and method efficiency in optimization," *Willey-Interscience, New York*, 1983.

[7] B. T. Polyak, *Introduction to Optimization*. Optimization Software New York, 1987.

[8] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87. Springer Science & Business Media, 2004.

[9] Y. Drori and M. Teboulle, "Performance of first-order methods for smooth convex minimization: a novel approach," *Mathematical Programming*, vol. 145, no. 1-2, pp. 451–482, 2014.

[10] Y. Drori, *Contributions to the Complexity Analysis of Optimization Algorithms*. PhD thesis, Tel-Aviv University, 2014.

[11] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Exact worst-case performance of first-order methods for composite convex optimization," *SIAM Journal on Optimization*, vol. 27, no. 3, pp. 1283–1313, 2017.

[12] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Smooth strongly convex interpolation and exact worst-case performance of first-order methods," *Mathematical Programming*, vol. 161, no. 1-2, pp. 307–345, 2017.

[13] 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*, 2016.

[14] A. B. Taylor, *Convex Interpolation and Performance Estimation of First-order Methods for Convex Optimization*. PhD thesis, Université catholique de Louvain, 2017.

[15] D. Kim and J. A. Fessler, "Optimized first-order methods for smooth convex minimization," *Mathematical Programming*, vol. 159, no. 1-2, pp. 81–107, 2016.

[16] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$)," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[17] Y. Drori, "The exact information-based complexity of smooth convex minimization," *Journal of Complexity*, 2016.

[18] D. Kim and J. A. Fessler, "Generalizing the optimized gradient method for smooth convex minimization," *preprint arXiv:1607.06764*, 2016.

[19] D. Kim and J. A. Fessler, "Another look at the "Fast Iterative Shrinkage/Thresholding Algorithm (FISTA)"," *preprint arXiv:1608.03861*, 2016.

[20] Y. Drori and M. Teboulle, "An optimal variant of kelley's cutting-plane method," *Mathematical Programming*, vol. 160, no. 1, pp. 321–351, 2016.

[21] L. Lessard, B. Recht, and A. Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 57–95, 2016.

[22] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Exact worst-case convergence rates of the proximal gradient method for composite convex minimization," *preprint arXiv:1705.04398*, 2017.

[23] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *Mathematical Programming*, vol. 146, no. 1-2, pp. 37–75, 2014.