A Tutorial on a Lyapunov-Based Approach to the Analysis of Iterative Optimization Algorithms

Bryan Van Scoy[1]   Laurent Lessard[2]

Abstract

Iterative gradient-based optimization algorithms are widely used to solve difficult or large-scale optimization problems. There are many algorithms to choose from, such as gradient descent and its accelerated variants such as Polyak's Heavy Ball method or Nesterov's Fast Gradient method. It has long been observed that iterative algorithms can be viewed as dynamical systems, and more recently, as robust controllers. Here, the "uncertainty" in the dynamics is the gradient of the function being optimized. Therefore, worst-case or average-case performance can be analyzed using tools from robust control theory, such as integral quadratic constraints (IQCs). In this tutorial paper, we show how such an analysis can be carried out using an alternative Lyapunov-based approach. This approach recovers the same performance bounds as with IQCs, but with the added benefit of constructing a Lyapunov function.

1 Introduction

In this paper, we consider unconstrained optimization problems of the form $\min_{x \in \mathbb{R}^d} f(x)$, where $f : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function.

Iterative gradient-based optimization algorithms attempt to solve such problems by starting from some initial guess $x_0 \in \mathbb{R}^d$ and iteratively updating x_k in an effort to converge to a local minimizer $x_\star \in \arg\min_x f(x)$. The simplest such algorithm is *gradient descent* (GD), which uses the update rule

$$x_{k+1} = x_k - \alpha \nabla f(x_k). \qquad \text{(GD)}$$

Intuitively, each step moves in the direction of the negative gradient (steepest descent direction). Here, α is the *stepsize*, which is a tunable parameter. Generally, a larger stepsize can result in faster convergence, but if the stepsize is too large, the algorithm may fail to converge.

---

[1] B. Van Scoy is with the Department of Electrical and Computer Engineering at Miami University, Oxford, OH 45056, USA bvanscoy@miamioh.edu

[2] L. Lessard is with the Department of Mechanical and Industrial Engineering at Northeastern University, Boston, MA 02115, USA l.lessard@northeastern.edu

Gradient descent may converge slowly when the function is poorly conditioned (when the condition number of the Hessian $\nabla^2 f$ is large). This is due to the fact that the contours of f are elongated so the iterates tend to oscillate in a non-productive manner. One way to alleviate this problem is by using *accelerated algorithms*. For example, Polyak's Heavy Ball (HB) [1, §3.2.1] uses an additional *momentum term* compared to (GD):

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}). \qquad \text{(HB)}$$

Alternatively, Nesterov's Fast Gradient (FG) [2, §2.2.1] evaluates the gradient at an interpolated point y_k:

$$\begin{aligned} x_{k+1} &= y_k - \alpha \nabla f(y_k), \\ y_{k+1} &= x_{k+1} + \beta(x_{k+1} - x_k). \end{aligned} \qquad \text{(FG)}$$

In Fig. 1, we compare the convergence of (GD), (HB), and (FG) on a simple quadratic function.
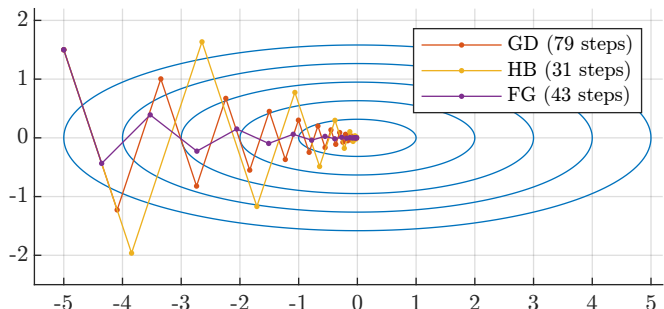
Fig. 1: Comparison of different iterative algorithms applied to $f(u, v) = u^2 + 10v^2$ (contour lines shown) with initial point $(-5, 1.5)$. Each algorithm is tuned to have the fastest possible convergence rate. Steps (in brackets) indicate the number of iterations needed to achieve $\|x_k - x_\star\| < 10^{-6}$. Gradient descent (GD) is less efficient than the Heavy Ball (HB) or Fast Gradient (FG) accelerated methods.

The tuning of an algorithm (the choice of α and β in the examples above) can have a dramatic effect on the convergence behavior. However, it is typically not feasible to find the optimal tuning, as this would depend on f, which presumably is a function complicated enough to warrant being minimized numerically. Instead, we seek performance guarantees over a *class of functions*. For example, we may know a priori that f is convex, or that it possesses a certain structural property, such as being quadratic. In the optimization literature, this sort of worst-case analysis is known as *algorithm analysis*.

1.1 The quadratic case

When f is a quadratic function as in least squares problems and as illustrated in Fig. 1, we can explicitly parameterize f, which makes the algorithm analysis straightforward. Specifically, $\nabla f(x) = Qx$ for some matrix Q. The dynamical systems for GD, HB, FG are therefore linear and convergence rate can be established via eigenvalue analysis. This leads to the following result.

Proposition 1 ([3, Prop. 1]). Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is quadratic and satisfies $0 \prec mI_d \preceq \nabla^2 f(x) \preceq LI_d$. The smallest ρ such that the iterates of the algorithms GD, HB, FG applied to f satisfy

$$\|x_k - x_\star\| \leq C\rho^k \|x_0 - x_\star\| \quad \text{for some } C > 0$$

is given by the following tunings (where $\kappa := L/m$)

	Optimal tuning	Rate bound
(GD)	$\alpha = \frac{2}{L+m}$	$\rho = \frac{\kappa-1}{\kappa+1}$
(HB)	$\alpha = \frac{4}{(\sqrt{L}+\sqrt{m})^2}, \beta = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2$	$\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$
(FG)	$\alpha = \frac{4}{3L+m}, \beta = \frac{\sqrt{3\kappa+1}-2}{\sqrt{3\kappa+1}+2}$	$\rho = \frac{\sqrt{3\kappa+1}-2}{\sqrt{3\kappa+1}}$

The tunings from Proposition 1 are the same as those used to generate the iterates shown in Fig. 1.

1.2 The smooth and strongly convex case

Another popular function class is the set of *smooth and strongly convex functions*, which we denote $\mathcal{F}_{m,L}$. These are continuously differentiable functions f that satisfy:

(i) L-Lipschitz gradients: $\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$ for all $x, y \in \mathbb{R}^d$.

(ii) m-strong convexity: $f(x) - \frac{m}{2}\|x\|^2$ is convex.

Here, we assume m and L can be estimated or are otherwise known a priori. This class of functions includes the quadratic functions from Proposition 1, but also includes non-quadratic functions. Functions of this type occur for example in regularized logistic regression or support vector machines with a smoothed hinge loss [4].

The class of smooth strongly convex functions cannot easily be parameterized, and the closed-loop dynamics of algorithms such as GD, HB, FG are generally nonlinear. Therefore, performing algorithm analysis requires a different approach from the quadratic case. Classical approaches typically involve clever manipulation of the inequalities that characterize Lipschitz gradients or strong convexity. One such approach is Nesterov's *estimating sequences* [2, §2.2.1], which yield the following result.

Proposition 2. Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is smooth and strongly convex with parameters $0 < m \leq L$. The method (FG) with parameters $\alpha = \frac{1}{L}$ and $\beta = \frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}$ achieves $\|x_k - x_\star\| \leq C\rho^k \|x_0 - x_\star\|$ for some $C > 0$, with $\rho = \sqrt{1 - \sqrt{\frac{m}{L}}}$.

1.3 Automated algorithm analysis

In recent years, there has been an effort to systematize, unify, and automate the process of algorithm analysis.

Drori and Teboulle [5] pioneered the idea that the worst-case performance of an algorithm (after a fixed number of iterations) could be characterized by a semidefinite program (SDP) whose size scaled with the number of iterations. It was later shown that this SDP could be convexified, leading to the so-called *Performance Estimation Problem* formulation [6].

Another popular approach is to express iterative methods as discretizations of gradient flows [7]. This view led to the unification of various accelerated algorithms, including (FG), and the derivation of an associated Lyapunov-based proof of convergence [8].

Yet another approach is to look for an asymptotic performance guarantee by leveraging tools from robust control [3]. Here, the idea is to view the algorithm as a Lur'e problem [9] (a linear time-invariant system in feedback with a static nonlinearity) and to adapt the integral quadratic constraint (IQCs) approach [10] to derive performance bounds. This approach also requires solving a convex SDP, but unlike PEP, its size is small and fixed (does not depend on the number of iterations).

1.4 Tutorial overview

In this tutorial, we will present an alternative Lyapunov-based approach to algorithm analysis, described in more detail in [11–13], that shares features in common with each of the aforementioned works. In the sections that follow, we will: 1) frame algorithm analysis as a Lur'e problem (as in the IQC approach); 2) use interpolation conditions to describe the set of smooth strongly convex functions (as in the PEP approach); 3) describe the lifting procedure we will use; 4) show how Lyapunov functions can be used to certify performance via solving convex SDPs; and 5) present numerical examples that illustrate the effectiveness of this approach for certifying both convergence rate and robustness to gradient noise.

2 Iterative algorithms as Lur'e problems

To express the iterative algorithm as a Lur'e problem, we begin by defining signals corresponding to the input y_k and output u_k of the gradient ∇f. For example, (FG) can be rewritten as

$$x_{k+1} = y_k - \alpha u_k,$$
$$y_{k+1} = x_{k+1} + \beta(x_{k+1} - x_k),$$
$$u_k = \nabla f(y_k).$$

Now define the augmented state $\xi_k := (x_k, x_{k-1})$ and put into standard state-space form to obtain

$$\xi_{k+1} = \begin{bmatrix} (1+\beta)I_d & -\beta I_d \\ I_d & 0_d \end{bmatrix} \xi_k + \begin{bmatrix} -\alpha I_d \\ 0_d \end{bmatrix} u_k, \quad (1a)$$

$$y_k = \begin{bmatrix} (1+\beta)I_d & -\beta I_d \end{bmatrix} \xi_k, \quad (1b)$$

$$u_k = \nabla f(y_k). \quad (1c)$$

To avoid the use of Kronecker products, it is convenient to express states, inputs, and outputs as *row vectors*, where there are d columns corresponding to the d dimensions of the original system. This transforms (1) into
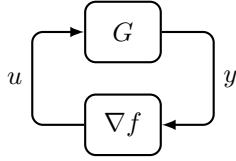
$$\xi_{k+1} = \begin{bmatrix} 1+\beta & -\beta \\ 1 & 0 \end{bmatrix} \xi_k + \begin{bmatrix} -\alpha \\ 0 \end{bmatrix} u_k, \qquad (2a)$$

$$y_k = \begin{bmatrix} 1+\beta & -\beta \end{bmatrix} \xi_k, \qquad (2b)$$

$$u_k = \nabla f(y_k), \qquad (2c)$$

where $\xi_k \in \mathbb{R}^{2\times d}$, $u_k \in \mathbb{R}^{1\times d}$, $y_k \in \mathbb{R}^{1\times d}$, and the gradient maps row vectors to row vectors: $\nabla f : \mathbb{R}^{1\times d} \to \mathbb{R}^{1\times d}$. Because of this convention, $\|\cdot\|$ denotes the Frobenius norm, for example, $\|\xi_k\|^2 := \sum_{i=1}^{2} \sum_{j=1}^{d} (\xi_k)_{ij}^2$.

We will use this convention from now on. Viewed as a block diagram, Eq. (2) has the following representation.

Here, G is a single-input single-output (SISO) system that depends on the algorithm, and it is understood that G acts separately on each of the d dimensions of the input. The system G has different realizations depending on the algorithm:

GD	HB	FG
$\left[\begin{array}{c\|c} 1 & -\alpha \\ \hline 1 & 0 \end{array}\right]$	$\left[\begin{array}{cc\|c} 1+\beta & -\beta & -\alpha \\ 1 & 0 & 0 \\ \hline 1 & 0 & 0 \end{array}\right]$	$\left[\begin{array}{cc\|c} 1+\beta & -\beta & -\alpha \\ 1 & 0 & 0 \\ \hline 1+\beta & -\beta & 0 \end{array}\right]$

The idea of representing iterative algorithms as dynamical systems is not new; see for example [1]. It is not restricted to the three algorithms above either; it can be used to represent distributed optimization algorithms [14], operator-splitting methods [12,15], and a variety of other numerical algorithms [16].

3   Lifted algorithm dynamics

Suppose the algorithm dynamics satisfy the state space equations (again using the row vector convention)

$$\xi_{k+1} = A\xi_k + B(u_k + w_k), \qquad (3a)$$

$$y_k = C\xi_k, \qquad (3b)$$

$$u_k = \nabla f(y_k). \qquad (3c)$$

In contrast with (2), we have included gradient noise w_k, which will allow us to analyze algorithm sensitivity in the sequel. In order to find the tightest possible performance bounds, we will search for a Lyapunov function that depends on a finite history of past algorithm iterates and

function values. To this end, define the *lifted* iterates

$$\boldsymbol{y}_k := \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-\ell} \end{bmatrix}, \quad \boldsymbol{u}_k := \begin{bmatrix} u_k \\ u_{k-1} \\ \vdots \\ u_{k-\ell} \end{bmatrix}, \quad \boldsymbol{f}_k := \begin{bmatrix} f_k \\ f_{k-1} \\ \vdots \\ f_{k-\ell} \end{bmatrix}. \qquad (4)$$

Also, define the truncation matrices Z_+, Z as follows.

$$Z_+ := \begin{bmatrix} I_\ell & 0_{\ell\times 1} \end{bmatrix} \quad \text{and} \quad Z := \begin{bmatrix} 0_{\ell\times 1} & I_\ell \end{bmatrix}. \qquad (5)$$

Multiplying a lifted iterate by Z on the left removes the most recent iterate (time k), while using Z_+ removes the oldest iterate (time $k-\ell$). Now define the lifted state

$$\boldsymbol{x}_k := \begin{bmatrix} \xi_k - \xi_\star \\ Z\boldsymbol{y}_k \\ Z\boldsymbol{u}_k \end{bmatrix} \in \mathbb{R}^{(n+2\ell)\times d}, \qquad (6)$$

which is the current state ξ_k and the ℓ previous inputs $u_{k-1}, \ldots, u_{k-\ell}$ and outputs $y_{k-1}, \ldots, y_{k-\ell}$ of the original system. The *lifted system* is a system with state $\boldsymbol{x}_k$, input (u_k, w_k), and output $(\boldsymbol{y}_k, \boldsymbol{u}_k)$ that is consistent with the original dynamics (3). These turn out to be

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{bmatrix} A & 0 & 0 \\ Z_+\boldsymbol{e}_1 C & Z_+ Z^\mathsf{T} & 0 \\ 0 & 0 & Z_+ Z^\mathsf{T} \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{x}_k + \underbrace{\begin{bmatrix} B \\ 0 \\ Z_+\boldsymbol{e}_1 \end{bmatrix}}_{\boldsymbol{B}} u_k + \underbrace{\begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{H}} w_k$$

$$\begin{bmatrix} \boldsymbol{y}_k \\ \boldsymbol{u}_k \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{e}_1 C & Z^\mathsf{T} & 0 \\ 0 & 0 & Z^\mathsf{T} \end{bmatrix}}_{\boldsymbol{C}} \boldsymbol{x}_k + \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{e}_1 \end{bmatrix}}_{\boldsymbol{D}} u_k, \qquad (7)$$

where $\boldsymbol{e}_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{\ell+1}$. We can recover the iterates of the original system (shifted to the fixed point) by projecting the augmented state and the input as

$$\tilde{\xi}_k = \underbrace{\begin{bmatrix} I_n & 0_{n\times(2\ell+1)} \end{bmatrix}}_{\boldsymbol{X}} \begin{bmatrix} \boldsymbol{x}_k \\ u_k \end{bmatrix}, \quad \tilde{y}_k = \underbrace{\begin{bmatrix} C & 0_{1\times(2\ell+1)} \end{bmatrix}}_{\boldsymbol{Y}} \begin{bmatrix} \boldsymbol{x}_k \\ u_k \end{bmatrix},$$

$$\text{and} \quad \tilde{u}_k = \underbrace{\begin{bmatrix} 0_{1\times(n+2\ell)} & 1 \end{bmatrix}}_{\boldsymbol{U}} \begin{bmatrix} \boldsymbol{x}_k \\ u_k \end{bmatrix}. \qquad (8)$$

Note that when $\ell = 0$, the lifted system reduces to the original system. That is, (7) becomes (3).

4   Interpolation Conditions

For the remainder of this paper, we will restrict our attention to smooth strongly convex functions, $f \in \mathcal{F}_{m,L}$.

The characterization of $\mathcal{F}_{m,L}$ from Section 1.2 depends on a continuum of points $x, y \in \mathbb{R}^d$. However, we will be analyzing the algorithm at a discrete set of iterates, so we will instead characterize $\mathcal{F}_{m,L}$ using *interpolation conditions*. These conditions were developed in [6] and provide necessary and sufficient conditions under which a given finite set of input-output data can be interpolated by a function in $f \in \mathcal{F}_{m,L}$.

Theorem 3 ([6, Thm. 4]). Consider the set $\{(y_k, u_k, f_k)\}$ for $k = 1, \ldots, m$. The following are equivalent.

(i) There exists a function $f \in \mathcal{F}_{m,L}$ satisfying

$$f(y_k) = f_k \quad \text{and} \quad \nabla f(y_k) = u_k \quad \text{for } k = 1, \ldots, m.$$

(ii) The following inequality holds for $i, j \in \{1, \ldots, m\}$.

$$2(L - m)(f_i - f_j) - mL\|y_i - y_j\|^2$$
$$+ 2(y_i - y_j)^{\mathsf{T}}(mu_i - Lu_j) - \|u_i - u_j\|^2 \geq 0.$$

Although we restrict our attention to smooth strongly convex functions, interpolation conditions can be derived for other function classes as well. See for example [6, 17].

Using our row vector convention, if we suppose that $y_k, u_k \in \mathbb{R}^{1 \times d}$, Item (ii) in Theorem 3 can be written as

$$q_{ij} := \mathrm{tr} \begin{bmatrix} y_i \\ y_j \\ u_i \\ u_j \end{bmatrix}^{\mathsf{T}} H \begin{bmatrix} y_i \\ y_j \\ u_i \\ u_j \end{bmatrix} + h^{\mathsf{T}} \begin{bmatrix} f_i \\ f_j \end{bmatrix} \geq 0, \quad \text{where} \quad (9)$$

$$H := \begin{bmatrix} -mL & mL & m & -L \\ mL & -mL & -m & L \\ m & -m & -1 & 1 \\ -L & L & 1 & -1 \end{bmatrix} \text{ and } h := 2(L-m)\begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Consider an algorithm in lifted form (7). We are interested in writing down as many valid inequalities as we can that relate its iterates. To this end, we will consider nonnegative linear combinations of the inequalities (9). The computation is presented in the following corollary.

Corollary 4. Consider a function $f \in \mathcal{F}_{m,L}$, and let $y_\star \in \mathbb{R}^{1 \times d}$ denote its optimizer, $u_\star = 0 \in \mathbb{R}^{1 \times d}$ the optimal gradient, and $f_\star \in \mathbb{R}$ the optimal function value. Let $y_k, \ldots, y_{k-\ell} \in \mathbb{R}^{1 \times d}$ be a sequence of iterates, and define $u_{k-i} := \nabla f(y_{k-i})$ and $f_{k-i} := f(y_{k-i})$ for $i = 0, \ldots, \ell$. Using these values, define the augmented vectors $\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{f}_k$ as in (4). Finally, define the index set $\mathcal{I} := \{1, 2, \ldots, \ell+1, \star\}$ and let $\boldsymbol{e}_i$ denote the ith unit vector in $\mathbb{R}^{\ell+1}$ with $\boldsymbol{e}_\star := 0 \in \mathbb{R}^{\ell+1}$. Then the inequality

$$\mathrm{tr} \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix}^{\mathsf{T}} \Pi(\Lambda) \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix} + \pi(\Lambda)^{\mathsf{T}} \boldsymbol{f}^t \geq 0 \qquad (10)$$

holds for all $\Lambda \in \mathbb{R}^{(\ell+2) \times (\ell+2)}$ such that $\Lambda_{ij} \geq 0$, where $\Pi(\Lambda)$ and $\pi(\Lambda)$ are defined in Eq. (11).

5 Lyapunov performance certification

In Propositions 1 and 2, we used *convergence rate* as a proxy for algorithm performance. For an algorithm G of the form (3) with fixed point $\xi_\star$, we define convergence rate formally as follows. Assume $w_k = 0$ for all k and let

$$\mathrm{RATE}(G) :=$$

$$\inf \left\{ r > 0 \;\middle|\; \sup_{f \in \mathcal{F}_{m,L}} \sup_{\xi_0 \in \mathbb{R}^{n \times d}} \sup_{k \geq 0} \frac{\|\xi_k - \xi_\star\|}{r^k \|\xi_0 - \xi_\star\|} < \infty \right\}.$$

A smaller ρ_G is desirable because it means that the algorithm is guaranteed to converge faster to its fixed point.

Another performance metric of interest is *sensitivity to additive gradient noise*. Suppose the noise inputs $w = (w_0, w_1, \ldots)$ are random, zero-mean, bounded variance, and independent across timesteps (not necessarily identically distributed). Specifically, $\mathbb{E}\, w_k = 0$, $\mathbb{E}\, w_k^{\mathsf{T}} w_k \preceq \sigma^2 I_d$, and $\mathbb{E}\, w_i^{\mathsf{T}} w_j = 0$ for all $i \neq j$. We denote the set of all such joint distributions over w as $\mathcal{P}_\sigma$.

For any fixed algorithm G, function $f \in \mathcal{F}_{m,L}$, initial point ξ_0, and noise distribution $w \sim \mathbb{P} \in \mathcal{P}_\sigma$, consider the stochastic iterate sequence $y_0, y_1, \ldots$ produced by G and let $y_\star := \arg\min_{y \in \mathbb{R}^d} f(y)$ be the unique minimizer of f. We define the noise sensitivity to be:

$$\mathrm{SENS}(G, \sigma^2) := \sup_{f \in \mathcal{F}_{m,L}} \sup_{\xi_0 \in \mathbb{R}^{n \times d}} \sup_{\mathbb{P} \in \mathcal{P}_\sigma}$$

$$\limsup_{T \to \infty} \sqrt{\mathbb{E}_{w \sim \mathbb{P}} \frac{1}{T} \sum_{k=0}^{T-1} \|y_k - y_\star\|^2}.$$

A smaller sensitivity is desirable because it means that the algorithm is more robust to gradient noise.

Both convergence rate and noise sensitivity can be computed exactly using eigenvalue analysis for the case of quadratic functions [11, 18]. For smooth strongly convex functions, IQC theory can instead be used to bound the rate and sensitivity [19].

We will now show an alternative Lyapunov approach, originally proposed in [11–13]. Specifically, we will use a Lyapunov function of the Lur'e–Postnikov form (quadratic in the state plus integral of the nonlinearity). For our lifted system representation, this takes the form

$$V(\boldsymbol{x}, \boldsymbol{f}) := \mathrm{tr}(\boldsymbol{x}^{\mathsf{T}} P \boldsymbol{x}) + p^{\mathsf{T}} Z \boldsymbol{f}, \qquad (12)$$

where P and p are parameters that must be optimized (P need not be positive definite), and the matrix Z is defined in (5). Both convergence rate and noise sensitivity can be verified by finding P and p such that the Lyapunov function (12) satisfies some algebraic conditions.

$$\Pi(\Lambda) := \sum_{i,j \in \mathcal{I}} \Lambda_{ij} \begin{bmatrix} -mL\,(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^{\mathsf{T}} & (\boldsymbol{e}_i - \boldsymbol{e}_j)(m\boldsymbol{e}_i - L\boldsymbol{e}_j)^{\mathsf{T}} \\ (m\boldsymbol{e}_i - L\boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^{\mathsf{T}} & -(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^{\mathsf{T}} \end{bmatrix}, \quad \pi(\Lambda) := 2\,(L - m) \sum_{i,j \in \mathcal{I}} \Lambda_{ij}\,(\boldsymbol{e}_i - \boldsymbol{e}_j) \quad (11)$$

Lemma 5. Consider the algorithm dynamics (3) and let V be of the form (12). If the iterates of the lifted system (7) with $w_k = 0$ satisfy the conditions

(i) $V(\boldsymbol{x}_k, \boldsymbol{f}_k) \geq \|\xi_k - \xi_\star\|^2$ and

(ii) $V(\boldsymbol{x}_{k+1}, \boldsymbol{f}_{k+1}) \leq r^2 V(\boldsymbol{x}_k, \boldsymbol{f}_k)$ for some $r > 0$,

then $\mathrm{RATE}(G) \leq r$.

Proof. Applying Items (i) and (ii) for $k \geq \ell$, we obtain

$$\|\xi_k - \xi_\star\|^2 \leq V(\boldsymbol{x}_k, \boldsymbol{f}_k) \leq \cdots \leq r^{2k} V(\boldsymbol{x}_0, \boldsymbol{f}_0)$$
$$\leq r^{2k}\big(C_0\|\xi_0 - \xi_\star\|^2 + C_1\big),$$

where C_0 and C_1 are constants that depend on the initialization of the algorithm and the parameters P, p. The result follows from applying the definition of $\mathrm{RATE}(G)$. ∎

Lemma 6. Consider the algorithm dynamics (3) and let V be of the form (12). If the iterates of the lifted system (7) satisfy the conditions

(i) $\mathbb{E}\, V(\boldsymbol{x}_k, \boldsymbol{f}_k) \geq 0$

(ii) $\mathbb{E}\, V(\boldsymbol{x}_{k+1}, \boldsymbol{f}_{k+1}) - \mathbb{E}\, V(\boldsymbol{x}_k, \boldsymbol{f}_k) + \mathbb{E}\, \|y_k - y_\star\|^2 \leq \gamma^2$ for some $\gamma > 0$,

then $\mathrm{SENS}(G, \sigma^2) \leq \gamma$.

Proof. applying Item (i) and averaging Item (ii) over $k = 0, \ldots, T-1$, we obtain

$$\frac{1}{T}\, \mathbb{E}\, \sum_{t=0}^{T-1} \|y_k - y_\star\|^2 \leq \frac{1}{T}\, \mathbb{E}\, V(\boldsymbol{x}_0, \boldsymbol{f}_0) + \gamma^2.$$

Taking the limit superior as $T \to \infty$ implies that γ is an upper bound on the senstivity to gradient noise. ∎

We will now show how Lemmas 5 and 6, together with the interpolation conditions of Section 4, can be used to efficiently certify convergence rate and noise sensitivity for iterative algorithm.

Theorem 7. Consider an algorithm G of the form (3) with with fixed point $(\xi_\star, y_\star, u_\star)$ applied to a function $f \in \mathcal{F}_{m,L}$. Suppose there is additive gradient noise with distribution in $\mathcal{P}_\sigma$. Define the truncation matrices in (5), the augmented state space and projection matrices in (7)–(8), and the valid inequality matrices in (11).

a) If there exist $P = P^{\mathsf{T}} \in \mathbb{R}^{(n+2\ell)\times(n+2\ell)}$ and $p \in \mathbb{R}^\ell$ and $\Lambda_1, \Lambda_2 \geq 0$ (elementwise) and $r > 0$ such that

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} P & 0 & 0 \\ 0 & -r^2 P & 0 \\ 0 & 0 & \Pi(\Lambda_1) \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \preceq 0 \quad (13a)$$

$$(Z_+ - r^2 Z)^{\mathsf{T}} p + \pi(\Lambda_1) \leq 0 \quad (13b)$$

$$\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X} + \begin{bmatrix} I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} -P & 0 \\ 0 & \Pi(\Lambda_2) \end{bmatrix} \begin{bmatrix} I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \preceq 0 \quad (13c)$$

$$-Z^{\mathsf{T}} p + \pi(\Lambda_2) \leq 0 \quad (13d)$$

then $\mathrm{RATE}(G) \leq r$.

b) If there exist $P = P^{\mathsf{T}} \in \mathbb{R}^{(n+2\ell)\times(n+2\ell)}$ and $p \in \mathbb{R}^\ell$ and $\Lambda_1, \Lambda_2 \geq 0$ (elementwise) such that

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} P & 0 & 0 \\ 0 & -P & 0 \\ 0 & 0 & \Pi(\Lambda_1) \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} + \boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y} \preceq 0 \quad (14a)$$

$$(Z_+ - Z)^{\mathsf{T}} p + \pi(\Lambda_1) \leq 0 \quad (14b)$$

$$\begin{bmatrix} I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} -P & 0 \\ 0 & \Pi(\Lambda_2) \end{bmatrix} \begin{bmatrix} I & 0 \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \preceq 0 \quad (14c)$$

$$-Z^{\mathsf{T}} p + \pi(\Lambda_2) \leq 0 \quad (14d)$$

then $\mathrm{SENS}(G, \sigma^2) \leq \sqrt{\sigma^2 d \cdot (\boldsymbol{H}^{\mathsf{T}} P \boldsymbol{H})}$.

Proof. Consider a trajectory (ξ_k, u_k, y_k, w_k) of the dynamics (3) with $w_k = 0$. Multiply (13a) and (13c) on the right and left by $(\boldsymbol{x}_k, u_k) \in \mathbb{R}^{n+\ell+1}$ and its transpose, respectively, and take the trace. Also, take the inner product of (13b) and (13d) with $\boldsymbol{f}_k$, which is valid because $\boldsymbol{f}_k$ is elementwise nonnegative. Next, sum the resulting (13a)+(13b) and (13c)+(13d), and the result immediately follows from Lemma 5 and Eq. (10).

For the second part of the proof, we do not restrict $w_k = 0$, and perform similar operations to the inequalities (14) as in the first part, this time taking expected values, and applying Lemma 6 and Eq. (10). ∎

5.1 Efficient numerical solutions

For fixed $r > 0$, the conditions (13) and (14) are linear matrix inequalities (LMIs) in the variables $P, p, \Lambda_1, \Lambda_2$. Each LMI has a size that depends on the number of algorithm states n and the lifting dimension ℓ, both of which are typically small. Critically, the size of the LMIs *does not* depend on d (the domain dimension of f), which can be very large in practice.

To find the best bound on $\mathrm{RATE}(G)$, one can perform a bisection search on r, at each step checking feasibility of Eq. (13). To find the best bound on $\mathrm{SENS}(G, \sigma^2)$, we can directly minimize $\boldsymbol{H}^{\mathsf{T}} P \boldsymbol{H}$ subject to Eq. (14).

6 Numerical examples

In Fig. 2, we plot bounds on $\mathrm{RATE}(G)$ for smooth strongly convex functions for different algorithms and choices of L/m, computed using Theorem 7. For all curves, a lifting dimension $\ell = 1$ was enough to get the best results. We tuned (GD) and (HB) as in Proposition 1, which recovers the result of [3, §4.6] whereby HB tuned in this manner may not be globally convergent in $\mathcal{F}_{m,L}$. We tuned (FG) as in Proposition 1, which yields a tighter rate bound that that of Proposition 2 (shown as FG*). Finally, we show the *Triple Momentum Method* [20], which has the fastest known convergence rate for this function class.

In Fig. 3, we plot sensitivity to gradient noise versus convergence rate for various algorithms applied to smooth strongly convex functions with $m = 1$ and $L = 8$,
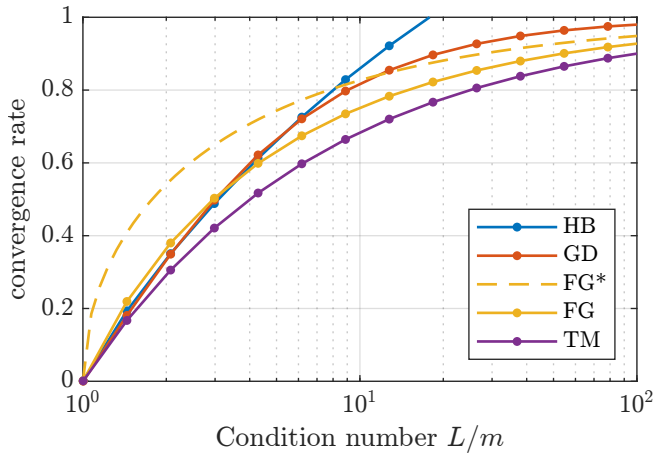
Fig. 2: Application of Theorem 7 to find bounds on RATE(G) for different algorithms applied to smooth strongly convex functions. See the text of Section 6 for details.

computed using Theorem 7. We used $\ell = 1$ for convergence rate and $\ell = 6$ for noise sensitivity. Also shown is GD*, which explores other tunings of GD with $0 \le \alpha \le \frac{2}{L}$ and shows that the regime $\alpha > \frac{2}{L+m}$ is always Pareto-suboptimal. Finally, we show RAM, which is the *Robust Accelerated Method* of [11] and achieves the best known trade-off between convergence rate and noise robustness.
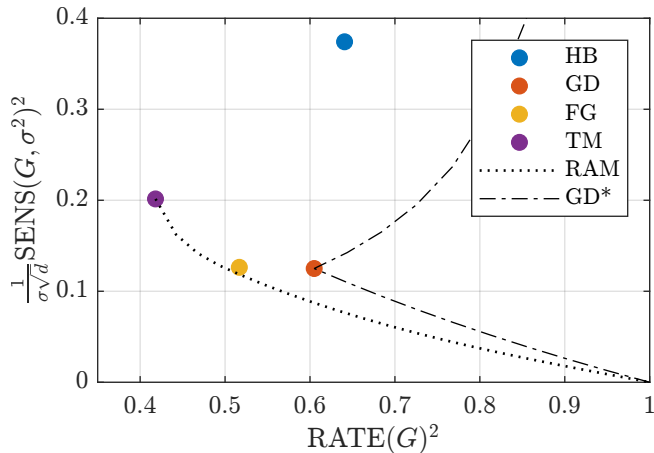
Fig. 3: Application of Theorem 7 to plot the trade-off between sensitivity to additive gradient noise and convergence rate in $\mathcal{F}_{1,8}$. See the text of Section 6 for details.

7   Concluding remarks

In this tutorial, we presented a Lyapunov-based approach for algorithm analysis, which is covered in more technical detail in [11–13].

Showcased in Figs. 2 and 3, this approach attains numerical results that empirically match those obtained using IQCs [3, 19], but does so using a familiar Lur'e–Postnikov Lyapunov function (12), and with results such as Theorem 7 that have straightforward proofs.

References

[1] B. T. Polyak, *Introduction to optimization*, ser. Translations series in mathematics and engineering.   Optimization Software, Inc., 1987.

[2] Y. Nesterov, *Lectures on convex optimization, second edition.* Springer, 2018, vol. 137.

[3] 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.

[4] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R.* Springer, 2017.

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

[6] 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.

[7] W. Su, S. Boyd, and E. Candes, "A differential equation for modeling nesterov's accelerated gradient method: theory and insights," *Advances in neural information processing systems*, vol. 27, 2014.

[8] A. C. Wilson, B. Recht, and M. I. Jordan, "A lyapunov analysis of accelerated methods in optimization," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 5040–5073, 2021.

[9] A. I. Lur'e and V. N. Postnikov, "On the theory of stability of control systems," *Applied mathematics and mechanics*, vol. 8, no. 3, pp. 246–248, 1944, in Russian.

[10] A. Megretski and A. Rantzer, "System analysis via integral quadratic constraints," *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 819–830, 1997.

[11] B. Van Scoy and L. Lessard, "The speed-robustness trade-off for first-order methods with additive gradient noise," *arXiv e-print*, 2021, http://arxiv.org/abs/2109.05059.

[12] L. Lessard, "The analysis of optimization algorithms: A dissipativity approach," *IEEE Control Systems Magazine*, vol. 42, no. 3, pp. 58–72, Jun. 2022.

[13] B. Van Scoy and L. Lessard, "Absolute stability via lifting and interpolation," in *IEEE Conference on Decision and Control*, 2022, pp. 6217–6223.

[14] A. Sundararajan, B. Van Scoy, and L. Lessard, "Analysis and design of first-order distributed optimization algorithms over time-varying graphs," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1597–1608, Dec. 2020.

[15] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan, "A general analysis of the convergence of ADMM," in *International Conference on Machine Learning*, Jul. 2015, pp. 343–352.

[16] A. Bhaya and E. Kaszkurewicz, *Control perspectives on numerical algorithms and matrix problems.*   SIAM, 2006.

[17] N. Bousselmi, J. M. Hendrickx, and F. Glineur, "Interpolation conditions for linear operators and applications to performance estimation problems," *arXiv e-print*, 2023, http://arxiv.org/abs/2302.08781.

[18] H. Mohammadi, M. Razaviyayn, and M. R. Jovanović, "Robustness of accelerated first-order algorithms for strongly convex optimization problems," *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2480–2495, 2021.

[19] S. Michalowsky, C. Scherer, and C. Ebenbauer, "Robust and structure exploiting optimisation algorithms: an integral quadratic constraint approach," *International Journal of Control*, vol. 94, no. 11, pp. 2956–2979, 2021.

[20] B. Van Scoy, R. A. Freeman, and K. M. Lynch, "The fastest known globally convergent first-order method for minimizing strongly convex functions," *IEEE Control Systems Letters*, vol. 2, no. 1, pp. 49–54, 2017.