The Speed–Robustness Trade-Off for First-Order Methods with Additive Gradient Noise

Bryan Van Scoy* Laurent Lessard†

Abstract

We study the trade-off between convergence rate and sensitivity to stochastic additive gradient noise for first-order optimization methods. Ordinary Gradient Descent (GD) can be made fast-and-sensitive or slow-and-robust by increasing or decreasing the stepsize, respectively. However, it is not clear how such a trade-off can be navigated when working with accelerated methods such as Polyak's Heavy Ball (HB) or Nesterov's Fast Gradient (FG) methods. We consider two classes of functions: (1) strongly convex quadratics and (2) smooth strongly convex functions. For each function class, we present a tractable way to compute the convergence rate and sensitivity to additive gradient noise for a broad family of first-order methods, and we present algorithm designs that trade off these competing performance metrics. Each design consists of a simple analytic update rule with two states of memory, similar to HB and FG. Moreover, each design has a scalar tuning parameter that explicitly trades off convergence rate and sensitivity to additive gradient noise. We numerically validate the performance of our designs by comparing their convergence rate and sensitivity to those of many other algorithms, and through simulations on Nesterov's "bad function".

1 Introduction

We consider the problem of designing robust first-order methods for unconstrained minimization. Given a continuously differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, consider solving the optimization problem

$$x^\star \in \operatorname*{arg\,min}_{x \in \mathbb{R}^d} f(x), \tag{1}$$

where the algorithm only has access to gradient measurements corrupted by *additive stochastic noise*.[1] Specifically, the algorithm can sample the oracle $g(x) := \nabla f(x) + w$, where w is zero-mean and independent across queries. This form of additive noise arises in various applications. For example, (i) to protect sensitive data, optimization algorithms may intentionally perturb the gradient by Gaussian noise in order to obtain differential privacy [6]; (ii) for some engineering systems, the gradient can only be obtained through noisy measurements [8]; and (iii) in risk minimization in the context of learning algorithms, the objective is to minimize the expectation of the loss function over the population distribution [29, 33, 36].

*B. Van Scoy is with the Department of Electrical and Computer Engineering, Miami University, Oxford, OH, USA. Email: bvanscoy@miamioh.edu

†L. Lessard is with the Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, USA. Email: l.lessard@northeastern.edu

[1] Preliminary versions of portions of this work appeared in the conference proceedings [56, 57].

Many iterative algorithms have been proposed to solve this problem, and most have tunable parameters. For example, Gradient Descent (GD) uses the update

$$\text{Gradient Descent (GD):} \qquad x^{t+1} = x^t - \alpha\, g(x^t), \tag{2}$$

where t is the iteration index and the stepsize α is a tunable parameter. Fig. 1 illustrates how the error $\|x^t - x^\star\|$ evolves under GD applied to strongly convex quadratic functions for different fixed α. The convergence of the error is characterized by an initial transient phase followed by a stationary phase. In the transient phase, the gradient dominates the noise, and the error converges at a linear rate. When the gradient is small enough that the noise becomes significant, the average error of the iterates converges to a constant value.

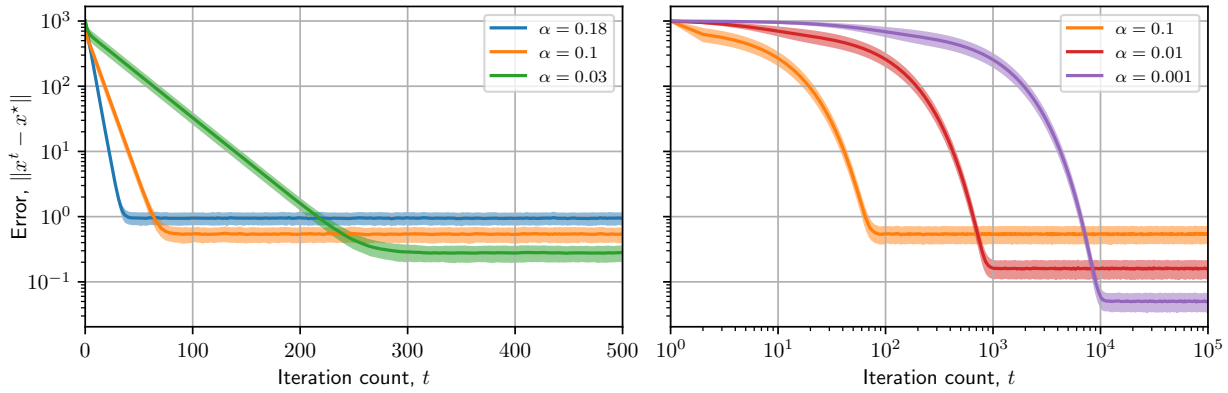

Figure 1: Trade-off between convergence rate and steady-state error (sensitivity to noise). Three different tunings of Gradient Descent (GD) with additive gradient noise are applied to random strongly convex quadratic functions on $\mathbb{R}^{10}$. Half of the Hessian eigenvalues are at $m = 1$, the other half at $L = 10$. The initialization is $x^0 = 1000\,\mathsf{e}_1$. Gradient noise is normally distributed $\mathcal{N}(0, I)$ and i.i.d. across iterations. The plot shows mean and ± 1 standard deviation of the error $\|x^t - x^\star\|$ for 1000 sample trajectories. On the right panel, iterations are plotted on a log scale.

This two-phase behavior is typical of stochastic methods.[2] The fundamental trade-off is that faster initial convergence comes at the cost of larger steady-state error.

Gradient Descent is easy to interpret and tune: the stepsize directly mediates the trade-off between convergence rate and sensitivity. Unfortunately, GD is generally slow to converge, and alternative methods can provide faster convergence rates. Two such methods are Polyak's Heavy Ball [47] and Nesterov's Fast Gradient [44], which use the updates

$$\text{Heavy Ball (HB):} \quad x^{t+1} = x^t - \alpha\, g(x^t) + \beta\,(x^t - x^{t-1}), \tag{3}$$

$$\text{Fast Gradient (FG):} \quad x^{t+1} = x^t - \alpha\, g\big(x^t + \beta\,(x^t - x^{t-1})\big) + \beta\,(x^t - x^{t-1}). \tag{4}$$

When using a noisy gradient oracle, these methods exhibit a stationary phase similar to that of GD in Fig. 1. A trade-off between convergence rate and sensitivity to noise must also exist for HB and FG, but there are now two parameters to tune, so it is unclear how they should be modified to mediate this trade-off.

[2] In the literature, stepsize is also known as *learning rate*. The transient phase is also known as the *search* or *burn-in* phase. The stationary phase is also known as the *convergence* or *steady-state* phase.

The goal of the present work is to study the trade-off between rate and sensitivity for first-order algorithms, and to design algorithms that trade off these competing performance metrics. We consider two well-studied classes of functions $f : \mathbb{R}^d \to \mathbb{R}$ characterized by scalar parameters m and L that satisfy $0 < m \leq L < \infty$.

- **Strongly convex quadratics ($Q_{m,L}$).** Functions such that, for some $f^\star \in \mathbb{R}$, $y^\star \in \mathbb{R}^d$, and $Q = Q^\mathsf{T} \in \mathbb{R}^{d\times d}$ with all eigenvalues in the interval $[m, L]$, the function has the form $f(y) = \frac{1}{2}(y - y^\star)^\mathsf{T} Q(y - y^\star) + f^\star$.
- **Smooth strongly convex functions ($F_{m,L}$).** Continuously differentiable functions for which $f(y) \geq f(x) + \nabla f(x)^\mathsf{T}(y - x) + \frac{m}{2}\|y - x\|^2$ (strongly convex) and $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ (Lipschitz gradients) for all $x, y \in \mathbb{R}^d$.

These classes of functions are nested in the sense that $Q_{m,L} \subseteq F_{m,L}$ with equality occurring when $m = L$. Moreover, all functions in these classes have a unique optimal point $y^\star = \arg\min_{y\in\mathbb{R}^d} f(y)$ satisfying $\nabla f(y^\star) = 0$.

For each function class, we design algorithms in the three-parameter family:

$$x^{t+1} = x^t - \alpha\, g\big(x^t + \eta\,(x^t - x^{t-1})\big) + \beta\,(x^t - x^{t-1}), \tag{5}$$

where α, β, η are scalar parameters. This parameterization was first introduced in [38] and is further discussed in Section 2.2. Note that GD, HB, FG are special cases of the three-parameter family (5) that use $\beta = \eta = 0$, $\eta = 0$, and $\eta = \beta$, respectively.

1.1 Main contributions

Our three main contributions are as follows.

1. For the function classes $Q_{m,L}$ and $F_{m,L}$, we present a method for efficiently bounding the worst-case convergence rate and sensitivity to additive gradient noise for a wide class of algorithms. The computational effort required to find the rate and sensitivity bounds for a given algorithm is independent of problem dimension d and takes fractions of a second on a laptop.
2. We present two new algorithms, RHB and RAM, for the function classes $Q_{m,L}$ and $F_{m,L}$, respectively. Each algorithm has the form (5), where (α, β, η) are explicit algebraic functions of m and L and a scalar tuning parameter r that directly trades off convergence rate and sensitivity to gradient noise.
3. We demonstrate through empirical studies that our algorithm designs effectively trade-off convergence rate and sensitivity to gradient noise. We use a brute-force approach to compare our algorithms against a dense sampling of algorithms of the form (5). We also show that our designs compare favorably to (i) popular algorithms such as nonlinear conjugate gradient and quasi-Newton methods, and (ii) existing designs that use more parameters.

Paper organization. We first provide additional background on the problem and describe our analysis and design methodology. In Section 2, we describe the problem setting and performance measures we will use. Sections 3 and 4 treat $Q_{m,L}$ and $F_{m,L}$, respectively. For each, we present a computationally tractable approach to computing convergence rate and noise sensitivity, and a

novel algorithm design. In Section 5, we present empirical studies that support the effectiveness of our designs and compare them to existing algorithms. We conclude in Section 6.

1.2 Literature review

Complexity bounds. The noise-free setting is well-studied, and algorithms have been discovered that achieve optimal worst-case iteration complexity for a variety of different function classes. We discuss these results in Sections 3.2 and 4.5, where we present our algorithm designs for $Q_{m,L}$ and $F_{m,L}$, respectively.

In the additive gradient noise setting, there are fundamental lower bounds on the asymptotic convergence rate of iterative methods. No matter what iterative scheme is used, $\mathbb{E}\|x^t - x^\star\|^2$ cannot decay asymptotically to zero faster than $1/t$. Roughly, this is because the rate at which error can decay is limited by the concentration properties of the gradient noise [29, 30]. This optimal asymptotic rate is achieved by Gradient Descent with a *diminishing stepsize* that decays as $1/t$ [9, Thm. 4.7] (GDDS). However, GDDS can suffer from poor finite-time performance: iterates may require many steps to reach the asymptotic regime. In practice, algorithms are run for a finite horizon, and if the transient phase persists, the resulting error can remain large. Hence, asymptotic optimality does not imply finite-time efficiency.

Stochastic approximation. In the additive gradient noise setting, a variety of techniques have been proposed to improve transient performance. For the case of least squares regression, a dramatic speedup can be achieved via careful manipulation of the stepsize or by using acceleration [27, 33]. Kulunchakov and Mairal [35] also show that any algorithm that converges linearly for smooth and strongly convex objectives in the noiseless setting can be converted into an algorithm that converges optimally (up to log factors) when additive noise is included.

Ghadimi and Lan [29] propose the AC-SA algorithm, which is a modification of Nesterov's Fast Gradient method that has near-optimal iteration complexity in the smooth and strongly convex setting. Besides characterizing the convergence of the average iterates, they also estimate the performance of the algorithm on a single problem instance. Building on this work, Cohen et al. [14] propose the algorithm AGD+, a "lazy" (dual averaging) counterpart of AC-SA for the smooth and convex setting, along with its extension μAGD+ for the strongly convex setting. In the strongly convex setting, μAGD+ improves on the convergence bound of the AC-SA algorithm, but does not achieve the lower complexity bound. In particular, applying [14, Corollary B.5] with $\gamma_i = c\sqrt{m/L}$ yields the bound

$$\frac{m}{2}\,\mathbb{E}\,\|y^t - y^\star\|^2 \;\le\; \left(1 - c\sqrt{\frac{m}{L}}\right)^t \frac{L-m}{2}\|x^0 - x^\star\|^2 + \frac{\sigma^2 c^2}{\sqrt{mL}}, \tag{6}$$

where we used the definition of strong convexity to bound the distance to optimality in terms of the optimality gap. The parameter $c \in (0, \sqrt{L/m}]$ can be chosen to trade off rate and sensitivity for this algorithm; however, we show that the resulting trade-off is strictly suboptimal (see Fig. 3).

None of the above algorithms achieve the optimal iteration complexity in the smooth strongly convex setting. One way to achieve the optimal complexity, however, is to exploit the rapid convergence of the transient phase by using a piecewise constant stepsize, effectively dividing the convergence into *epochs* or *stages*. This can be done on a predetermined schedule, or by using a statistical test

to detect the phase transition which then triggers the parameter change [12, 14]. In Ghadimi and Lan's companion paper [30], for instance, they propose a multi-stage version of AC-SA that achieves the optimal expected rate of convergence (up to constants) in this setting. Aybat et al. [4] build on Nesterov's Accelerated Stochastic Gradient method to construct the Multi-Stage Accelerated Stochastic Gradient method (M-ASG). This algorithm simultaneously achieves optimal iteration complexity (again, up to constants) in both deterministic and stochastic cases without knowledge of the noise parameters.

The aforementioned multi-stage algorithms tune the single-stage algorithm stepsizes and the restart schedule to achieve desirable transient and asymptotic convergence, and they achieve the optimal complexity up to constants. In the present work work, we focus on analyzing and designing single-stage algorithms with fixed stepsizes that can be tuned via a single scalar parameter to explicitly trade-off convergence rate and sensitivity to noise. In other words, our algorithms can easily be adjusted to be made faster (and more sensitive to noise), or more robust to noise (and slower). When tuned to be as fast as possible, our algorithm designs recover algorithms that are known to converge with the optimal linear rate in the noise-free setting for the respective function classes (see Section 5). Our algorithm designs may be used to converge at a linear rate to a predetermined noise level, or the trade-off parameter may be adjusted over time to construct multi-stage variants[3].

Other noise models. While we restrict our attention to additive stochastic gradient noise in the present work, other inexact oracles have been studied in the literature. When the gradient noise is due to sampling a finite-sum objective function, viable strategies include incremental gradient or variance reduction methods such as SVRG [34], SAGA [18], and many others [1, 25, 45, 58, 61].

A prominent alternative model is to assume *deterministic*[4] bounded noise, which may be additive or multiplicative. For instance, Devolder et al. [21] propose an inexact deterministic first-order oracle and show that, under this oracle, acceleration necessarily results in an accumulation of gradient errors in the unconstrained smooth (not strongly convex) case. For this setting, the same authors propose the Intermediate Gradient Method, a family of first-order methods that can be tuned to trade off convergence rate and sensitivity to gradient errors [20]. The same authors also extended this oracle to analyze inexact first-order methods in the strongly convex case [19]; there is still a trade-off between rate and sensitivity, but errors no longer accumulate and can converge linearly to within a constant ball about the optimal solution. Multiplicative deterministic noise has been studied in [38], for which the Robust Momentum (RM) [15] was designed to trade off convergence rate and sensitivity. Stochastic Gradient Descent has also been analyzed under a hybrid additive and multiplicative deterministic oracle [32].

1.3 Methodology

We now provide an overview of our analysis and design framework and point to related techniques. Our method for bounding the convergence rate r and sensitivity γ for the class $F_{m,L}$ relies on solving a small linear matrix inequality (LMI). This idea builds upon several related works.

The *performance estimation problem* (PEP) framework [23, 52] uses LMIs to directly search for a

[3] While we do not propose a restart mechanism or time-varying parameter schedule in the present work, we illustrate such an approach in Section 5.3 through a hand-tuned schedule.

[4] Also known as *worst-case* or *adversarial* noise.

problem instance that achieves the worst-case performance. The PEP framework has been successfully applied to numerous algorithms, such as the proximal gradient method [53], operator splitting methods [48], and gradient descent using an exact line search with noisy search directions [16]. Of particular interest, [50] uses PEP to analyze iterative algorithms under various noise models when the objective is smooth and convex. This approach searches for a time-varying potential function whose existence certifies a performance bound. To obtain closed-form bounds, increasingly large LMIs are solved and their numerical solutions guide the construction of explicit expressions for the potential function parameters as functions of the iteration index. Alternatively, small PEPs can be formulated and solved to search for potential functions with fixed parameters as in [17].

Viewing algorithms as discrete-time dynamical systems, Integral Quadratic Constraints (IQCs) from control theory [26, 38, 42] may be used to search for worst-case performance guarantees. This approach also leads to LMIs that characterize *asymptotic* performance, although the ensuing performance bounds are not tight in general.

Finally, LMIs may be used to directly search for a *Lyapunov function*, which is a generalized notion of "energy" stored in the system. If a fraction of the energy dissipates at each iteration, this is akin to proving convergence at a specified rate [31, 37, 54]. Similar to IQCs, Lyapunov functions provide asymptotic (albeit more interpretable) performance guarantees.

In the present work, we adopt a Lyapunov approach most similar to [54], but we generalize it to include both convergence rate and sensitivity to noise. We also explain in Section 4.6 how the Lyapunov approach is related to IQCs.

Given an LMI that establishes the performance (convergence rate or sensitivity to noise) of a given first-order method, the next step is to *design* algorithms that exploit this trade-off. Several approaches have been proposed: i) One can parameterize a family of candidate algorithms and search over parameters to effectively trade off convergence rate and sensitivity to noise. Such a problem is typically non-convex, so one must resort to exhaustive search [38] or nonlinear numerical solvers that find local optima. ii) Using convex relaxations or other heuristics such as coordinate descent, the algorithm design problem can be solved approximately. While this approach may lead to conservative designs, it has the benefit of being automated, flexible, and amenable to efficient convex solvers [42]. iii) In certain settings, the controller parameters can be eliminated from the LMI entirely, yielding bounds that hold for all algorithms. This approach has been used to show that the Triple Momentum (TM) method achieves the optimal worst-case rate over the class $F_{m,L}$ [39, 49].

As in the first approach, we use the parameterized family of algorithms (5) and search over the parameters. However, our approach to design is algebraic rather than numeric. We find analytic solutions to the non-convex semidefinite programs that arise when the algorithm parameters are treated as decision variables. This approach enables us to find explicit analytic expressions for our algorithm parameters. Nevertheless, we still make use of numerical approaches in order to validate our choice of parameterization and the efficacy of our designs, and to compare the performance of our designs to that of existing algorithms; see Section 5.

2 Problem setting and assumptions

To solve the optimization problem (1), we consider iterative first-order methods described by dynamics of the form

$$\xi^{t+1} = A\xi^t + B(u^t + w^t), \qquad y^t = C\xi^t, \qquad u^t = \nabla f(y^t), \tag{7}$$

where $\xi^t \in \mathbb{R}^{n\times d}$ is the *state* of the algorithm, $y^t \in \mathbb{R}^{1\times d}$ is where we evaluate the gradient, $u^t \in \mathbb{R}^{1\times d}$ is the (exact) gradient, $w^t \in \mathbb{R}^{1\times d}$ is the noise, and t is the *iteration*. The state is the *memory* of the algorithm; its size reflects the number of past iterates that must be stored at each timestep. Given an algorithm (A, B, C), objective function f, and initial condition ξ^0, a *trajectory* is any sequence $(\xi^t, u^t, y^t, w^t)_{t\geq 0}$ that satisfies (7). This general framework encompasses a wide variety of fixed-parameter first-order iterative methods; we discuss this in more detail in Section 2.2.

Remark 1 (important notational convention)**.** *Throughout this paper, we represent iterates as* row vectors*, so the matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, and $C \in \mathbb{R}^{1\times n}$ act on the* columns *of the iterates while the gradient oracle $\nabla f : \mathbb{R}^{1\times d} \to \mathbb{R}^{1\times d}$ acts on the* rows*. Thus, $\|\cdot\|$ denotes the Frobenius norm, for example, $\|\xi^t\|^2 := \sum_{i=1}^n \sum_{j=1}^d (\xi_{ij}^t)^2$.*

A necessary condition for (27) to represent a valid iterative algorithm is that there exists a fixed point corresponding to the stationary point of the objective function. In other words, there should exist $\xi^\star \in \mathbb{R}^{n\times d}$ and $y^\star \in \mathbb{R}^{1\times d}$ satisfying $\xi^\star = A\xi^\star$ and $y^\star = C\xi^\star$. Taking this condition columnwise, we obtain the following.

Assumption 1 (algorithm form)**.** *The matrix A has an eigenvalue at 1, and the associated eigenvector $v \in \mathbb{R}^n$ satisfies $Cv \neq 0$.*[5]

2.1 Performance evaluation

Let $\mathcal{A} = (A, B, C)$ denote an algorithm as in (7) and let $\mathcal{F} \in \{Q_{m,L}, F_{m,L}\}$ denote one of function classes defined in Section 1.

Convergence rate. The convergence rate describes the first phase of convergence observed in Fig. 1: exponential decrease of the error. In this regime, gradients are relatively large compared to the noise, so we set $w^t = 0$ for all $t \geq 0$. For any algorithm $\mathcal{A}$ with initial point ξ^0 and fixed point $\xi^\star$ and any function $f \in \mathcal{F}$, consider the trajectory $(\xi^0, \xi^1, \dots)$ produced by $\mathcal{A}$. We define the linear convergence rate as

$$\textsc{rate}(\mathcal{A}, \mathcal{F}) := \inf \left\{ r > 0 \;\middle|\; \sup_{f\in\mathcal{F}} \sup_{\xi^0\in\mathbb{R}^{n\times d}} \sup_{t\geq 0} \frac{\|\xi^t - \xi^\star\|}{r^t \|\xi^0 - \xi^\star\|} < \infty \right\}. \tag{8}$$

If the convergence rate $r = \textsc{rate}(\mathcal{A}, \mathcal{F})$ is finite, then for all $\varepsilon > 0$, there exists $c > 0$ such that the trajectory satisfies $\|\xi^t - \xi^\star\| \leq c(r+\varepsilon)^t \|\xi^0 - \xi^\star\|$ for all functions $f \in \mathcal{F}$, initial points $\xi^0 \in \mathbb{R}^{n\times d}$, and iterations $t \geq 0$. This definition corresponds to the conventional notion of *linear convergence rate* used in the worst-case analysis of deterministic algorithms. If $r < 1$, the algorithm is said to be *globally linearly convergent*, and for all $\varepsilon > 0$, we have $\|y^t - y^\star\| = O((r+\varepsilon)^t)$. Smaller r corresponds to a faster (worst-case) convergence rate.

[5] In the language of control theory, the discrete-time system (A, B, C) has an *observable integrator*.

Sensitivity. The sensitivity characterizes the steady-state phase of convergence observed in Fig. 1. The steady-state error depends on the noise characteristics. We make the following assumptions on the noise sequence.

Assumption 2 (Noise sequence)**.** *We assume that the noise sequence $w^0, w^1, \dots$ has joint distribution $\mathbb{P} \in \mathcal{P}_\sigma$, with parameter σ to be defined shortly. We assume the set of admissible joint distributions $\mathcal{P}_\sigma$ satisfies:*

1. Independence across time. *For all $\mathbb{P} \in \mathcal{P}_\sigma$, if $w \sim \mathbb{P}$, then w^t and w^τ are independent for all $t \neq \tau$. Then we may characterize the joint distribution $\mathbb{P} \in \mathcal{P}_\sigma$ by its associated marginal distributions $(\mathbb{P}^0, \mathbb{P}^1, \dots)$. We do not assume the $\mathbb{P}^t$ are necessarily identical.*
2. Zero-mean and bounded variance. *For all $(\mathbb{P}^0, \mathbb{P}^1, \dots) \in \mathcal{P}_\sigma$, we have that $\mathbb{E}_{w^t \sim \mathbb{P}^t}(w^t) = 0$ and $\mathbb{E}_{w^t \sim \mathbb{P}^t}(\|w^t\|^2) \leq \sigma^2$.*

Assumption 2 implies $\mathcal{P}_\sigma$ is completely characterized by the variance bound σ^2. For algorithm $\mathcal{A}$, function class $\mathcal{F}$, initial ξ^0, and family of distributions $\mathcal{P}_\sigma$, consider the stochastic iterate sequence $y^0, y^1, \dots$ produced by $\mathcal{A}$. We define noise sensitivity as

$$\textsc{sens}(\mathcal{A}, \mathcal{F}) := \sup_{f \in \mathcal{F}} \sup_{\xi^0 \in \mathbb{R}^{n \times d}} \sup_{\mathbb{P} \in \mathcal{P}_\sigma} \limsup_{T \to \infty} \sqrt{\mathbb{E}_{w \sim \mathbb{P}} \frac{1}{\sigma^2 T} \sum_{t=0}^{T-1} \|y^t - y^\star\|^2}. \tag{9}$$

For all cases we consider, the normalized definition of sensitivity (9) does not depend on σ. A smaller sensitivity is desirable because it means the algorithm achieves small error in spite of gradient perturbations. This definition is similar to that used in recent works exploring first-order algorithms using techniques from robust control [5, 42, 43].

Remark 2. *Some previous works have defined sensitivity with respect to the squared-norm of the state $\|\xi^t - \xi^\star\|^2$ [42,43]. This quantity, however, depends on the state-space realization; performing the coordinate transformation $(A, B, C) \mapsto (TAT^{-1}, TB, CT^{-1})$ for some invertible matrix T does not change the sequence of inputs u^t or outputs y^t, but it transforms the states: $\xi^t \mapsto T\xi^t$. Our definition of sensitivity* (9) *is invariant under such transformations.*

2.2 Algorithm parameterization

While our *analysis* results apply to the general model (7), for *design* we will further restrict the class of algorithms to those with state dimension $n = 2$. Algorithms of the form (7) have $n^2 + 2n$ degrees of freedom in the matrices A, B, C, so the case $n = 2$ should require 8 parameters. However, many of these parameters are redundant, and under Assumption 1, it turns out the case $n = 2$ is completely characterized by the three-parameter family (5).

Lemma 2.1 (three-parameter family)**.** *Under Assumption 1, any algorithm of the form* (7) *with state dimension $n = 2$ is equivalent to an algorithm in the three-parameter family* (5) *for some (α, β, η). By equivalent, we mean that both algorithms produce the same sequence of iterates $(y^0, y^1, \dots)$ given appropriate initialization.*

Proof. See Section A.1 ∎

We refer to specific algorithms from the three-parameter family (5) by their parameters (α, β, η). All such algorithms satisfy Assumption 1. For specific values for the parameters, the algorithm (5)

recovers several known algorithms, shown in Table 1, which will serve as useful benchmarks for our designs. Later in Section 5.2, we provide numerical evidence that further justifies our choice of parameterization.

Table 1: Comparison of different algorithms with their recommended/standard tunings. For RM, the parameter satisfies $1 - \sqrt{\frac{m}{L}} \leq r \leq 1 - \frac{m}{L}$, and interpolates between TM and GD with $\alpha = \frac{1}{L}$.

Algorithm name	α	β	η
Gradient Descent (GD)	$\frac{1}{L}$ or $\frac{2}{L+m}$	0	0
Heavy Ball (HB), [47]	$\frac{4}{(\sqrt{L}+\sqrt{m})^2}$	$\left(\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}\right)^2$	0
Fast Gradient (FG), [44]	$\frac{1}{L}$	$\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}$	$\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}$
Triple Momentum (TM), [55]	$\frac{2\sqrt{L}-\sqrt{m}}{L^{3/2}}$	$\frac{(\sqrt{L}-\sqrt{m})^2}{L+\sqrt{mL}}$	$\frac{(\sqrt{L}-\sqrt{m})^2}{2L-m+\sqrt{mL}}$
Robust Momentum (RM), [15]	$\frac{(1-r)^2(1+r)}{m}$	$\frac{Lr^3}{L-m}$	$\frac{mr^3}{(L-m)(1-r)^2(1+r)}$

Different triples (α, β, η) generally correspond to different algorithms, with one important exception: Gradient Descent has a degenerate family of parameterizations.

Proposition 2.2. *Gradient Descent with parameterization $(\alpha, 0, 0)$ can also be parameterized by $\left(\alpha(1-\beta),\ \beta,\ \frac{\beta}{1-\beta}\right)$ for any choice of $\beta \neq 1$.*

To see why this holds, substitute $(\alpha, \beta, \eta) \mapsto \left(\alpha(1-\beta), \beta, \frac{\beta}{1-\beta}\right)$ into (5), and observe that the update becomes GD applied to the quantity $y^t := \frac{1}{1-\beta}(x^t - \beta x^{t-1})$.

Remark 3. *Theorems 2.1 and 2.2 can also be proved using the notion of a* transfer function *(TF) [2, §2.7], which is the linear map from the z-transform of $(u^t + w^t)$ to the z-transform of y^t. The TF of (7) is $G(z) = C(zI - A)^{-1}B$ and when two dynamical systems have the same TF, they are equivalent in the sense of Theorem 2.1. When $n = 2$, $G(z)$ is a rational function with one zero and two poles, one of which must be at $z = 1$ by Assumption 1. This leaves three degrees of freedom (the second pole, the zero, and a constant gain). Similarly, it is easy to check that the TF for $\left(\alpha(1-\beta),\ \beta,\ \frac{\beta}{1-\beta}\right)$ is $G(z) = \frac{-\alpha}{z-1}$, which is independent of β.*

In the next two sections, we focus on $Q_{m,L}$ and $F_{m,L}$. For each, we provide a tractable approach for bounding the convergence rate and sensitivity, and we design algorithms of the form (5) that effectively trade off these performance metrics.

3 Strongly convex quadratic functions

3.1 Performance bounds for $Q_{m,L}$

Quadratics have been treated extensively in recent works. For this class, the convergence rate has been characterized [38], and closed-form expressions for the sensitivity of GD, HB, and FG have been obtained [5, 43]. We now present versions of these results adapted to our algorithm class of interest.

Lemma 3.1 ($Q_{m,L}$ analysis)**.** *Consider algorithm $\mathcal{A} = (A, B, C)$ defined in (7) satisfying Assumption 1 applied to $f \in Q_{m,L}$ defined in Section 1. Assume the noise sequence satisfies Assumption 2.*

Algorithm $\mathcal{A}$ has convergence rate

$$\textsc{rate}(\mathcal{A}, Q_{m,L}) = \sup_{q\in[m,L]} \rho\big(A + qBC\big),$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix (largest eigenvalue magnitude). If $\textsc{rate}(\mathcal{A}, Q_{m,L}) < 1$*, the algorithm has finite sensitivity given by*

$$\textsc{sens}(\mathcal{A}, Q_{m,L}) = \sup_{q\in[m,L]} \sqrt{B^\mathsf{T} P_q B},$$

where P_q is the unique solution to the linear equation

$$(A + qBC)^\mathsf{T} P_q (A + qBC) - P_q + C^\mathsf{T} C = 0. \tag{10}$$

Proof. See Section A.2. ∎

The linear equation (10) is a *Lyapunov equation*, and the fact that it has a unique solution is provided in [24, §11.4.4]. Similar results have appeared in the context of algorithm analysis for quadratics [5, 43] and use that the sensitivity is equivalent to the $\mathcal{H}_2$ norm of the system, which can be computed using a Lyapunov approach.

The expressions for the rate and sensitivity in Theorem 3.1 may be difficult to evaluate if the matrices (A, B, C) are large[6]. Fortunately, the sizes of these matrices only depend on the state dimension n, which is typically small ($n \leq 2$ for all methods in Table 1). Moreover, the dimension d of the function domain does not affect the complexity of the convergence rate or sensitivity formula.

3.2 Algorithm design for $Q_{m,L}$

For strongly convex quadratic functions, first-order methods can achieve exact convergence in d iterations when there is no gradient noise, where d is the dimension of the domain of f. One such example is the Conjugate Gradient (CG) method [46, Thm. 5.4]. However, when the number of iterations t satisfies $t < d$, exact convergence is not possible in general. Nesterov's lower bound [44, Thm. 2.1.13] demonstrates that for any $t \geq 0$, one can construct a function $f \in Q_{m,L}$ with domain dimension $d > t$ such that:

$$\|y^t - y^\star\| \geq \left(\frac{\sqrt{L} - \sqrt{m}}{\sqrt{L} + \sqrt{m}}\right)^t \|y^0 - y^\star\|. \tag{11}$$

This lower bound holds for any first-order method such that y^t is a linear combination of y^0 and (the exact) past gradients $\nabla f(y^0), \ldots, \nabla f(y^{t-1})$. This class includes not only CG but also methods with unbounded memory.

In the regime $t < d$, the CG method matches Nesterov's lower bound [46, Thm. 5.5] and is therefore optimal in terms of worst-case rate. However, it is not clear how CG should be adjusted to be robust in the presence of additive gradient noise, since it has no tunable parameters.

[6] Neither $\rho(A + qBC)$ nor P_q are convex functions of q in general.

The HB method (3), when tuned as in Table 1, also matches Nesterov's lower bound when applied to the function class $Q_{m,L}$ [47, §3.2.1], but has a simpler implementation than CG: its updates are linear and its parameters are constant.

We adopted the three-parameter class (α, β, η) described in Section 2.2 as our search space for optimized algorithms because HB belongs to this class and achieves optimal performance on $Q_{m,L}$ when there is no noise. Substituting the three-parameter algorithm (27) into Theorem 3.1, we obtain the following result.

Corollary 3.2 ($Q_{m,L}$ analysis, reduced)**.** *Consider the three-parameter algorithm $\mathcal{A} = (\alpha, \beta, \eta)$ defined in* (5) *applied to a strongly convex quadratic $f \in Q_{m,L}$ defined in Section 1, and assume the noise sequence satisfies Assumption 2. The algorithm has convergence rate*

$$\textsc{rate}(\mathcal{A}, Q_{m,L}) = \max_{q \in \{m,L\}} \begin{cases} \sqrt{\beta - \alpha\eta q} & \text{if } \Delta < 0, \\ \frac{1}{2}\left(|\beta + 1 - \alpha q - \alpha\eta q| + \sqrt{\Delta}\right) & \text{if } \Delta \geq 0, \end{cases}$$
$$\text{where } \Delta := (\beta + 1 - \alpha q - \alpha\eta q)^2 - 4(\beta - \alpha\eta q). \quad (12)$$

If $\textsc{rate}(\mathcal{A}, Q_{m,L}) < 1$*, the algorithm has sensitivity*

$$\textsc{sens}(\mathcal{A}, Q_{m,L}) = \max_{q \in \{m,L\}} \sqrt{h(q)} := \sqrt{\frac{\alpha(1 + \beta + (1 + 2\eta)\alpha\eta q)}{q(1 - \beta + \alpha\eta q)(2 + 2\beta - (1 + 2\eta)\alpha q)}} \quad (13)$$

Proof. See Section A.3. ∎

In Theorem 3.2, the suprema from Theorem 3.1 are replaced by a simple maximum; we only need to check the endpoints of the interval $[m, L]$.

We begin with a baseline; the convergence and sensitivity of GD on $Q_{m,L}$.

Theorem 3.3 (GD analysis for $Q_{m,L}$)**.** *Consider the function class $Q_{m,L}$ and let r be a parameter chosen with $\frac{L-m}{L+m} \leq r < 1$. Then, under Assumption 2, the algorithm $\mathcal{A}$ of the form* (5) *with $\alpha = \frac{1}{m}(1 - r)$, $\beta = 0$, and $\eta = 0$ achieves*

$$\textsc{rate}(\mathcal{A}, Q_{m,L}) = r, \qquad \text{and} \qquad \textsc{sens}(\mathcal{A}, Q_{m,L}) = \frac{1}{m}\sqrt{\frac{1 - r}{1 + r}}.$$

Proof. Omitted; follows from straightforward algebra. ∎

Our proposed algorithm for the class $Q_{m,L}$ is a special tuning of Heavy Ball, which we named *Robust Heavy Ball* (RHB). The RHB algorithm was found by careful analysis of the analytic expressions for the rate and sensitivity in Theorem 3.2. The algorithm is described in the following theorem.

Theorem 3.4 (Robust Heavy Ball, RHB)**.** *Consider the function class $Q_{m,L}$ and let r be a parameter chosen with $\frac{\sqrt{L} - \sqrt{m}}{\sqrt{L} + \sqrt{m}} \leq r < 1$. Then under Assumption 2, the algorithm $\mathcal{A}$ of the form* (5) *with $\alpha = \frac{1}{m}(1 - r)^2$, $\beta = r^2$, and $\eta = 0$ achieves*

$$\textsc{rate}(\mathcal{A}, Q_{m,L}) = r \qquad \text{and} \qquad \textsc{sens}(\mathcal{A}, Q_{m,L}) = \frac{1}{m}\sqrt{\frac{1 - r^4}{(1 + r)^4}}.$$

Proof. We first show that the parameter r is in fact the convergence rate of the algorithm. Substituting the algorithm parameters $\alpha = \frac{1}{m}(1-r)^2$, $\beta = r^2$, and $\eta = 0$ into (12), we obtain $\Delta = -(1-r)^4\left(\frac{q}{m}-1\right)\left(\left(\frac{1+r}{1-r}\right)^2 - \frac{q}{m}\right)$. Rearranging the inequalities $m \le q \le L$ and $\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}} \le r < 1$ yields $1 \le \frac{q}{m} \le \frac{L}{m} \le \left(\frac{1+r}{1-r}\right)^2 < \infty$. Thus, we conclude that $\Delta \le 0$ and we have $\textsc{rate}(\mathcal{A}, Q_{m,L}) = \sqrt{\beta} = r$, as required. Now substitute the parameters into (13) and obtain

$$h(m) - h(L) = \frac{(1-r)(r^2+1)(\frac{L}{m}-1)\left(\left(\frac{1+r}{1-r}\right)^2 - \frac{L}{m}\right)}{Lm(r+1)^3\left(1+\left(\frac{1+r}{1-r}\right)^2 - \frac{L}{m}\right)} \ge 0,$$

so $h(q)$ is maximized when $q = m$, and $\textsc{sens}(\mathcal{A}, Q_{m,L}) = \sqrt{h(m)} = \frac{1}{m}\sqrt{\frac{1-r^4}{(1+r)^4}}$. ∎

Although we set out to design an algorithm in the three-parameter class (α, β, η), our designed algorithm RHB uses $\eta = 0$, so it is a particular tuning of HB. Our numerical experiments in Section 5.1 suggest that this parameter choice yields the most effective trade-off between rate and sensitivity. In other words, it is unnecessary to use a nonzero η when optimizing over the class $Q_{m,L}$.

If we choose the smallest possible $r = \frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}$ (fastest possible convergence rate), then we recover Polyak's tuning of HB, whose convergence rate matches Nesterov's lower bound. It is straightforward to check that the sensitivity is a monotonically decreasing function of r, so as the convergence rate slows down (r increases), the algorithm becomes less sensitive to noise. Thus, r is a single tunable parameter that enables us to trade off convergence rate with sensitivity to noise.

4 Smooth strongly convex functions

We now consider the class $F_{m,L}$ of strongly convex functions whose gradient is Lipschitz continuous. In contrast to the $Q_{m,L}$ case, the function class $F_{m,L}$ is not readily parameterizable. Therefore, we use a Lyapunov approach to certify performance bounds.

4.1 Lyapunov analysis

Our analysis is based on searching for a function that certifies either a particular convergence rate (assuming no noise) or level of sensitivity (assuming noise). When used in the context of certifying stability of dynamical systems, such a function is called a *Lyapunov* function. This function will depend on the state $\boldsymbol{x}^t \in X$ of a (to-be-defined) dynamical system. In Section 4.2, we will show how to construct this augmented system from the algorithm (7). Before doing so, we first show how to use such a function to certify performance.

Bounds on the rate of convergence To bound the rate of convergence of an algorithm, we search for a function $V(\boldsymbol{x})$ that decreases along trajectories and is lower bounded by the squared norm of the iterates.

Lemma 4.1 (Lyapunov analysis for rate of convergence)**.** *Consider an algorithm $\mathcal{A} = (A, B, C)$ defined in* (7) *satisfying Assumption 1 applied to a function $f \in \mathcal{F}$ with no gradient noise ($w^t = 0$*

for all t). If there exists a function $V : X \to \mathbb{R}$ and a constant $r > 0$ such that, for all functions $f \in \mathcal{F}$ and all iterations $t \geq 0$, the iterates $\boldsymbol{x}^t$ satisfy the conditions

(i) Lower bound condition: $V(\boldsymbol{x}^t) \geq \|\xi^t - \xi^\star\|^2$ *and*

(ii) Decrease condition: $V(\boldsymbol{x}^{t+1}) \leq r^2\, V(\boldsymbol{x}^t)$,

then $\textsc{rate}(\mathcal{A}, \mathcal{F}) \leq r$.

Proof. By applying the lower bound condition followed by the decrease condition at each iteration $t \geq 0$, we obtain the chain of inequalities $\|\xi^t - \xi^\star\|^2 \leq V(\boldsymbol{x}^t) \leq r^2\, V(\boldsymbol{x}^{t-1}) \leq \ldots \leq r^{2t}\, V(\boldsymbol{x}^0)$. From Eq. (8), we obtain $\textsc{rate}(\mathcal{A}, \mathcal{F}) \leq r$, as required. ∎

Bounds on the sensitivity to noise To bound the sensitivity of an algorithm to noise, we search for a function $V(\boldsymbol{x})$ that satisfies the following conditions.

Lemma 4.2 (Lyapunov analysis for sensitivity to noise)**.** *Consider an algorithm $\mathcal{A} = (A, B, C)$ defined in* (7) *satisfying Assumption 1 applied to a function $f \in \mathcal{F}$ with additive gradient noise satisfying Assumption 2. If there exists a function $V : X \to \mathbb{R}$ and a constant $\gamma > 0$ such that, for all functions $f \in \mathcal{F}$ and all iterations $t \geq 0$, the iterates $\boldsymbol{x}^t$ satisfy the conditions*

(i) Lower bound condition: $\mathbb{E}\, V(\boldsymbol{x}^t) \geq 0$ *and*

(ii) Decrease condition: $\mathbb{E}\, V(\boldsymbol{x}^{t+1}) - \mathbb{E}\, V(\boldsymbol{x}^t) + \mathbb{E}\|y^t - y^\star\|^2 \leq \sigma^2\gamma^2$,

then $\textsc{sens}(\mathcal{A}, \mathcal{F}) \leq \gamma$.

Proof. Averaging the decrease condition over $t = 0, \ldots, T-1$, we obtain

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

Applying the lower bound condition to the first term and taking the limit superior as $T \to \infty$, the second term vanishes and the result follows from (9). ∎

In both cases (Theorems 4.1 and 4.2), we will see that searching over functions $V(\boldsymbol{x})$ of a particular form can be cast as a semidefinite program.

4.2 Lifted dynamics

To obtain performance bounds, we use a *lifting* approach that searches for certificates of performance that depend on a finite history of past algorithm iterates and function values. The main idea is to lift the state to a higher dimension so that we can search over a broader class of certificates to reduce the conservativeness of the bound. We denote the *lifting dimension* by $\ell \geq 0$, which dictates the dimension of the lifted state, and we use boldface symbols to denote quantities related to the lifted dynamics.

Given a trajectory of the system (7), we define the following augmented vectors, each consisting of

$\ell + 1$ consecutive iterates of the system:

$$\boldsymbol{y}^t := \begin{bmatrix} y^t - y^\star \\ \vdots \\ y^{t-\ell} - y^\star \end{bmatrix}, \qquad \boldsymbol{u}^t := \begin{bmatrix} u^t \\ \vdots \\ u^{t-\ell} \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{f}^t := \begin{bmatrix} f^t - f^\star \\ \vdots \\ f^{t-\ell} - f^\star \end{bmatrix}, \tag{14}$$

where $\boldsymbol{y}^t, \boldsymbol{u}^t \in \mathbb{R}^{(\ell+1)\times d}$ and $\boldsymbol{f}^t \in \mathbb{R}^{\ell+1}$ with $f^t := f(y^t)$ and $f^\star := f(y^\star)$ (recall our convention that algorithm inputs and outputs u^t, y^t are *row* vectors). Also, define the truncation matrices $Z, Z_+ \in \mathbb{R}^{\ell\times(\ell+1)}$ as

$$Z_+ := \begin{bmatrix} I_\ell & 0_{\ell\times 1} \end{bmatrix} \quad \text{and} \quad Z := \begin{bmatrix} 0_{\ell\times 1} & I_\ell \end{bmatrix}. \tag{15}$$

Multiplying an augmented vector on the left by Z removes the most recent iterate at time t, while multiplication by Z_+ removes the last iterate at time $t - \ell$. Using these augmented vectors, we then define the augmented state

$$\boldsymbol{x}^t := (\boldsymbol{\xi}^t, Z\boldsymbol{f}^t), \qquad \text{where} \qquad \boldsymbol{\xi}^t := \begin{bmatrix} \xi^t - \xi^\star \\ Z\boldsymbol{y}^t \\ Z\boldsymbol{u}^t \end{bmatrix} \in \mathbb{R}^{(n+2\ell)\times d}, \tag{16}$$

which consists of the deviations of the state ξ^t and past inputs $y^{t-1}, \dots, y^{t-\ell}$, outputs $u^{t-1}, \dots, u^{t-\ell}$, and function values $f^{t-1}, \dots, f^{t-\ell}$ of the original system from equilibrium. The augmented dynamics, which follow from Eqs. (7) and (14) to (16), are

$$\boldsymbol{\xi}^{t+1} = \underbrace{\begin{bmatrix} A & 0 & 0 \\ Z_+\mathsf{e}_1 C & Z_+ Z^\mathsf{T} & 0 \\ 0 & 0 & Z_+ Z^\mathsf{T} \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{\xi}^t + \underbrace{\begin{bmatrix} B \\ 0 \\ Z_+\mathsf{e}_1 \end{bmatrix}}_{\boldsymbol{B}} u^t + \underbrace{\begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{H}} w^t, \tag{17a}$$

$$\begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix} = \underbrace{\begin{bmatrix} \mathsf{e}_1 C & Z^\mathsf{T} & 0 \\ 0 & 0 & Z^\mathsf{T} \end{bmatrix}}_{\boldsymbol{C}} \boldsymbol{\xi}^t + \underbrace{\begin{bmatrix} 0 \\ \mathsf{e}_1 \end{bmatrix}}_{\boldsymbol{D}} u^t, \tag{17b}$$

where $\mathsf{e}_1 = (1, 0, \dots, 0) \in \mathbb{R}^{\ell+1}$. We can recover the iterates of the original system by projecting the augmented state and the input as

$$\xi^t - \xi^\star = \underbrace{\begin{bmatrix} I_n & 0_{n\times(2\ell+1)} \end{bmatrix}}_{\boldsymbol{X}} \begin{bmatrix} \boldsymbol{\xi}^t \\ u^t \end{bmatrix}, \; y^t - y^\star = \underbrace{\begin{bmatrix} C & 0_{1\times(2\ell+1)} \end{bmatrix}}_{\boldsymbol{Y}} \begin{bmatrix} \boldsymbol{\xi}^t \\ u^t \end{bmatrix}, \; u^t = \underbrace{\begin{bmatrix} 0_{1\times(n+2\ell)} & 1 \end{bmatrix}}_{\boldsymbol{U}} \begin{bmatrix} \boldsymbol{\xi}^t \\ u^t \end{bmatrix}. \tag{18}$$

Remark 4 (State reduction for noise-free case)**.** *When there is no noise ($w^t = 0$), the component $\boldsymbol{\xi}^t$ of the augmented state* (16) *has linearly dependent rows; knowledge of the past state $\xi^{t-\ell}$ and subsequent inputs $u^{t-\ell}, \dots, u^{t-1}$ is enough to reconstruct the outputs $y^{t-\ell}, \dots, y^{t-1}$. Thus, in this case, we could work with a smaller lifted state that does not include outputs. Such a reduction does not change the results of the analysis, but makes the semidefinite programs smaller and therefore (slightly) more computationally efficient.*

4.3 Interpolation conditions

A useful characterization of the function class $F_{m,L}$ is given by the *interpolation conditions*. These are necessary and sufficient conditions on a sequence of points to be interpolable by a function in the class. We state the result from [52, Thm. 4], rephrased to match our notation.

Proposition 4.3 (Interpolation conditions for $F_{m,L}$)**.** *Let $y^1, \dots, y^k \in \mathbb{R}^{1\times d}$ and $u^1, \dots, u^k \in \mathbb{R}^{1\times d}$ and $f^1, \dots, f^k \in \mathbb{R}$. The following statements are equivalent.*

1) There exists $f \in F_{m,L}$ such that $f(y^i) = f^i$ and $\nabla f(y^i) = u^i$ for $i = 1, \dots, k$.

2) For all $i, j \in \{1, \dots, k\}$, the following inequality holds:

$$\operatorname{tr}\big((u^i)^\mathsf{T}(y^i - y^j)\big) - (f^i - f^j) + \frac{1}{2\,(L-m)} \operatorname{tr} \begin{bmatrix} y^i - y^j \\ u^i - u^j \end{bmatrix}^\mathsf{T} \begin{bmatrix} -mL & m \\ m & -1 \end{bmatrix} \begin{bmatrix} y^i - y^j \\ u^i - u^j \end{bmatrix} \geq 0. \quad (19)$$

We now develop a version of Theorem 4.3 with a single parameterized inequality that holds for the augmented vectors defined in (14).

Lemma 4.4. *Consider an algorithm (7) applied to a function $f \in F_{m,L}$. Define the augmented iterates in (14) and the index set $I := \{1, \dots, \ell+1, \star\}$, and let e_i denote the i^{th} unit vector in $\mathbb{R}^{\ell+1}$ with $\mathsf{e}_\star := 0 \in \mathbb{R}^{\ell+1}$. Then the inequality*

$$\operatorname{tr} \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix}^\mathsf{T} \Pi(\Lambda) \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix} + \pi(\Lambda)^\mathsf{T} \boldsymbol{f}^t \geq 0 \quad (20)$$

holds for all $\Lambda \in \mathbb{R}^{(\ell+2)\times(\ell+2)}$ such that $\Lambda \geq 0$ (elementwise), where

$$\Pi(\Lambda) := \sum_{i,j\in I} \Lambda_{ij} \begin{bmatrix} -mL\,(\mathsf{e}_i - \mathsf{e}_j)(\mathsf{e}_i - \mathsf{e}_j)^\mathsf{T} & (\mathsf{e}_i - \mathsf{e}_j)(m\mathsf{e}_i - L\mathsf{e}_j)^\mathsf{T} \\ (m\mathsf{e}_i - L\mathsf{e}_j)(\mathsf{e}_i - \mathsf{e}_j)^\mathsf{T} & -(\mathsf{e}_i - \mathsf{e}_j)(\mathsf{e}_i - \mathsf{e}_j)^\mathsf{T} \end{bmatrix}, \quad (21a)$$

$$\pi(\Lambda) := 2\,(L-m) \sum_{i,j\in I} \Lambda_{ij}\,(\mathsf{e}_i - \mathsf{e}_j). \quad (21b)$$

Proof. For each $i \in I$, define the vectors $\tilde{y}_i := \mathsf{e}_i^\mathsf{T} \boldsymbol{y}^t$ and $\tilde{u}_i := \mathsf{e}_i^\mathsf{T} \boldsymbol{u}^t$ and $\tilde{f}_i := \mathsf{e}_i^\mathsf{T} \boldsymbol{f}^t$. By definition, the points $(\tilde{y}_i, \tilde{u}_i, \tilde{f}_i)$ are interpolated by the function $f \in F_{m,L}$, so by Theorem 4.3 the interpolation conditions (19) are satisfied. The proof is then completed by noting that the inequality (20) can be expanded as

$$\sum_{i,j\in I} \Lambda_{ij} \Big(-mL\|\tilde{y}_i - \tilde{y}_j\|^2 + 2\,(\tilde{y}_i - \tilde{y}_j)(m\tilde{u}_i - L\tilde{u}_j)^\mathsf{T} - \|\tilde{u}_i - \tilde{u}_j\|^2 + 2(L-m)(\tilde{f}_i - \tilde{f}_j) \Big) \geq 0,$$

which is a nonegative weighted combination of the interpolation conditions, where the interpolation condition between index i and j is scaled by $2\,(L-m)\Lambda_{ij} \geq 0$. ∎

4.4 Performance bounds for $F_{m,L}$

We now use the interpolation conditions to search for a Lyapunov function that depends on the state $\boldsymbol{x}^t$ of the lifted system that satisfies the conditions in Section 4.1 to certify either the convergence rate or sensitivity. Motivated by the fact that the nonnegative inequalities in (20) are quadratic in the inputs and outputs of the gradient and linear in the function values, we search for certificates $V : \mathbb{R}^{(n+2\ell)\times d} \times \mathbb{R}^\ell \to \mathbb{R}$ of the form

$$V(\boldsymbol{x}) := \operatorname{tr}\big(\boldsymbol{\xi}^\mathsf{T} P \boldsymbol{\xi}\big) + p^\mathsf{T} \boldsymbol{f}, \tag{22}$$

where $\boldsymbol{x} = (\boldsymbol{\xi}, \boldsymbol{f})$ is the state of the lifted system, and we search over parameters $P = P^\mathsf{T} \in \mathbb{R}^{(n+2\ell)\times(n+2\ell)}$ and $p \in \mathbb{R}^\ell$. Since V is quadratic in the augmented state and linear in the augmented function values, we can efficiently search for such Lyapunov functions using the following linear matrix inequalities that leverage the characterization of smooth strongly convex functions in Theorem 4.4.

Theorem 4.5 ($F_{m,L}$ analysis)**.** *Consider an algorithm $\mathcal{A} = (A, B, C)$ defined in* (7) *satisfying Assumption 1 applied to a function $f \in F_{m,L}$ defined in Section 1 with additive gradient noise satisfying Assumption 2. Define the matrices in Eqs.* (15)*,* (17)*,* (18) *and* (21)*.*

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

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \end{bmatrix}^\mathsf{T} \begin{bmatrix} P & 0 \\ 0 & -r^2 P \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^\mathsf{T} \Pi(\Lambda_1) \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \preceq 0 \tag{23a}$$

$$(Z_+ - r^2 Z)^\mathsf{T} p + \pi(\Lambda_1) \leq 0 \tag{23b}$$

$$\boldsymbol{X}^\mathsf{T} \boldsymbol{X} - \begin{bmatrix} I & 0 \end{bmatrix}^\mathsf{T} P \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^\mathsf{T} \Pi(\Lambda_2) \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \preceq 0 \tag{23c}$$

$$-Z^\mathsf{T} p + \pi(\Lambda_2) \leq 0 \tag{23d}$$

then $\textsc{rate}(\mathcal{A}, F_{m,L}) \leq r$*.*

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

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \end{bmatrix}^\mathsf{T} \begin{bmatrix} P & 0 \\ 0 & -P \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ I & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^\mathsf{T} \Pi(\Lambda_1) \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} + \boldsymbol{Y}^\mathsf{T} \boldsymbol{Y} \preceq 0 \tag{24a}$$

$$(Z_+ - Z)^\mathsf{T} p + \pi(\Lambda_1) \leq 0 \tag{24b}$$

$$-\begin{bmatrix} I & 0 \end{bmatrix}^\mathsf{T} P \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^\mathsf{T} \Pi(\Lambda_2) \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \preceq 0 \tag{24c}$$

$$-Z^\mathsf{T} p + \pi(\Lambda_2) \leq 0 \tag{24d}$$

then $\textsc{sens}(\mathcal{A}, F_{m,L}) \leq \sqrt{\boldsymbol{H}^\mathsf{T} P \boldsymbol{H}}$*.*

Proof. See Section A.4. ∎

Remark 5. *When the lifting dimension ℓ is zero, the lifted system is identical to the original system. In other words, $\xi^t = \boldsymbol{\xi}^t$. The system matrices also satisfy $A = \boldsymbol{A}$, and similarly for B. As ℓ is increased, the LMIs* (23) *and* (24) *have the potential to yield less conservative bounds on the rate and sensitivity, respectively.*

Both bounds for the class $F_{m,L}$ in Theorem 4.5 can be evaluated and optimized efficiently. The sizes of the LMIs depend only on n and ℓ, which are typically small.

4.5 Algorithm design for $F_{m,L}$

For the case with no noise, the Triple Momentum (TM) method [55] attains the fastest-known worst-case convergence rate of $1 - \sqrt{\frac{m}{L}}$ over the function class $F_{m,L}$. Recent work by Drori and Taylor [22, 51] has confirmed that this rate is in fact optimal.

We adopted the three-parameter class (α, β, η) described in Section 2.2 as our search space for optimized algorithms, because it includes TM as a special case, and FG, which is a popular choice for this function class. We begin with the GD baseline.

Theorem 4.6 (GD analysis for $F_{m,L}$)**.** *Consider the function class $F_{m,L}$ and let r be a parameter chosen with $\frac{L-m}{L+m} \leq r < 1$. Then, under Assumption 2, the algorithm $\mathcal{A}$ of the form* (5) *with $\alpha = \frac{1}{m}(1-r)$, $\beta = 0$, and $\eta = 0$ achieves*

$$\textsc{rate}(\mathcal{A}, Q_{m,L}) = r, \qquad \textit{and} \qquad \textsc{sens}(\mathcal{A}, Q_{m,L}) = \frac{1}{m}\sqrt{\frac{1-r}{1+r}}.$$

Proof. See Section A.5. ∎

GD achieves the same worst-case rate and sensitivity in $F_{m,L}$ (Theorem 4.6) as in $Q_{m,L}$ (Theorem 3.3). This confirms the common knowledge that quadratics are worst-case function for GD.

Our proposed algorithm, the *Robust Accelerated Method* (RAM), uses a parameter r to trade off convergence rate and sensitivity to noise. To design RAM, we applied the procedure outlined in Section 1.3 to the rate LMI (23) with lifting dimension $\ell = 1$. Our main result is Theorem 4.7, which provides the exact convergence rate of RAM. While we do not construct a bound on the sensitivity, we show in Section 5 that it effectively trades off rate and sensitivity. To obtain a numerical bound on the sensitivity for a given parameter r, one can use the analysis result in Theorem 4.5.

Theorem 4.7 (Robust Accelerated Method, RAM)**.** *Consider the function class $F_{m,L}$, and let r be a parameter chosen with $1 - \sqrt{\frac{m}{L}} \leq r < 1$. Then, the algorithm $\mathcal{A}$ of the form* (5) *with tuning*

$$\alpha = \frac{(1+r)(1-r)^2}{m}, \quad \beta = r\,\frac{L\,(1-r+2r^2) - m\,(1+r)}{(L-m)(3-r)}, \quad \eta = r\,\frac{L\,(1-r^2) - m\,(1+2r-r^2)}{(L-m)(3-r)(1-r^2)}$$

achieves the performance $\textsc{rate}(\mathcal{A}, F_{m,L}) = r$.

Proof. See Section A.6. ∎

Remark 6. *When r is set to its minimum value of $1 - \sqrt{\frac{m}{L}}$, the Robust Accelerated Method in Theorem 4.7 reduces to the Triple Momentum Method (see Table 1).*

4.6 Comparison with IQCs from robust control

We now compare our analysis in Theorem 4.5 for computing the worst-case convergence rate over $F_{m,L}$ with the use of integral quadratic constraints (IQCs) [41]. Specifically, we apply the upper bound on convergence rate from the LMI in [38, Eq. 3.8] with the weighted off-by-one IQC in [38, Lemma 10]. The result is summarized below.

Proposition 4.8 (Weighted off-by-one IQC)**.** *Consider the algorithm $\mathcal{A} = (A, B, C)$ defined in* (7) *satisfying Assumption 1 applied to a function $f \in F_{m,L}$ defined in Section 1. If there exists $Q \succ 0$ satisfying*

$$\begin{bmatrix} \hat{A}^\mathsf{T} Q \hat{A} - r^2 Q & \hat{A}^\mathsf{T} Q \hat{B} \\ \hat{B}^\mathsf{T} Q \hat{A} & \hat{B}^\mathsf{T} Q \hat{B} \end{bmatrix} + \begin{bmatrix} \hat{C} & \hat{D} \end{bmatrix}^\mathsf{T} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{C} & \hat{D} \end{bmatrix} \preceq 0,$$

then $\textsc{rate}(\mathcal{A}, F_{m,L}) \leq r$*, where the matrices $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ are given by*

$$\hat{A} = \begin{bmatrix} A & 0 \\ -LC & 0 \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} B \\ 1 \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} LC & r^2 \\ -mC & 0 \end{bmatrix}, \quad \hat{D} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

If $Q \succ 0$ satisfies the weighted off-by-one IQC LMI from Theorem 4.8, then we can construct a feasible point for our analysis LMI from (23) with $\ell = 1$ as follows.

$$p = 2\,(L - m), \quad \Lambda_1 = \begin{bmatrix} 0 & 0 & 0 \\ r^2 & 0 & 0 \\ 1 - r^2 & 0 & 0 \end{bmatrix}, \quad \Lambda_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \text{and} \tag{25a}$$

$$P = \begin{bmatrix} A & B \\ -LC & 1 \end{bmatrix}^\mathsf{T} Q \begin{bmatrix} A & B \\ -LC & 1 \end{bmatrix} - \begin{bmatrix} LmC^\mathsf{T} C & -mC^\mathsf{T} \\ -mC & 1 \end{bmatrix}. \tag{25b}$$

In this case, a Lyapunov function that certifies the linear rate r is

$$V(\xi^t, u^t, f^t) = \begin{bmatrix} \xi^{t+1} \\ \zeta^{t+1} \end{bmatrix}^\mathsf{T} Q \begin{bmatrix} \xi^{t+1} \\ \zeta^{t+1} \end{bmatrix} + 2\,(L - m) \left(f^t - \frac{m}{2} \|y^t\|^2 \right) - \|u^t - m y^t\|^2, \tag{26}$$

where $\xi^{t+1} = A\xi^t + Bu^t$ and $\zeta^{t+1} := u^t - Ly^t$. Therefore, using the weighted off-by-one IQC can be interpreted as searching over this restricted class of Lyapunov functions. Even though our analysis for computing the convergence rate in Theorem 4.5 is more general, the weighted off-by-one IQC formulation appears to be general enough to prove tight results. For example, while RAM was designed using the more general analysis, its Lyapunov function has the special form (26).

In [42], Zames–Falb multipliers [60] (of which the weighted off-by-one IQC is a special case) are used to formulate LMIs for computing both the convergence rate and the sensitivity. Just as with the weighted off-by-one IQC, using general Zames–Falb multipliers can also be interpreted as searching over a restricted class of Lyapunov functions, although a detailed comparison is beyond the scope of this work.

While the weighted off-by-one IQC formulation appears to achieve tight bounds on the convergence rate, computing tight bounds on the sensitivity requires the more general LMI (24); see Section 5.4 for further discussion.

5 Numerical validation

We perform numerical experiments to verify that our designs (i) effectively trade off convergence rate and noise sensitivity, (ii) use an adequate number of parameters (they are neither under- nor over-parameterized), and (iii) outperform popular iterative schemes when applied to a worst-case test function. Our code is available at `https://github.com/QCGroup/optalg`.

5.1 Empirical validation

To empirically validate our designs from Theorems 3.4 and 4.7, we performed a brute-force analysis of algorithms (α, β, η) and made a scatter plot of sensitivity vs. convergence rate. The following result facilitates sampling by providing bounds on admissible (α, β, η).

Lemma 5.1 (parameter restriction)**.** *Consider a three-parameter algorithm $\mathcal{A} = (\alpha, \beta, \eta)$ defined in Eq. (5). Let $\mathcal{F} \in \{Q_{m,L}, F_{m,L}\}$. If* $\textsc{rate}(\mathcal{A}, \mathcal{F}) < 1$*, then:*

$$0 < \alpha < \frac{4}{L}, \qquad \frac{-2}{L-m} < \alpha\eta < \frac{2}{L-m}, \qquad \begin{cases} -1 + L(\alpha\eta) < \beta < 1 + m(\alpha\eta) & \text{if } \alpha\eta \geq 0 \\ -1 + m(\alpha\eta) < \beta < 1 + L(\alpha\eta) & \text{if } \alpha\eta < 0. \end{cases}$$

Proof. See Section A.7. ∎

From Theorem 5.1, we see that α, $\alpha\eta$, and β each have finite ranges. A convenient way to grid the space of possible (α, β, η) values is to first grid over α, then $\alpha\eta$, then β, in a nested fashion, extracting associated (α, β, η) values at each step. Due to the multiplicative nature of the parameter α, we opted to sample α logarithmically in the range $\left[10^{-5}, \frac{4}{L}\right]$, but to sample $\alpha\eta$ and β linearly in their associated intervals.

Strongly convex quadratics $(Q_{m,L})$. We show our brute-force search for the class $Q_{m,L}$ in Fig. 2 for $Q_{1,10}$ and $Q_{1,100}$. For this figure, we used the sampling approach based on Theorem 5.1 with $500 \times 201 \times 200$ samples for α, $\alpha\eta$, and β, respectively.

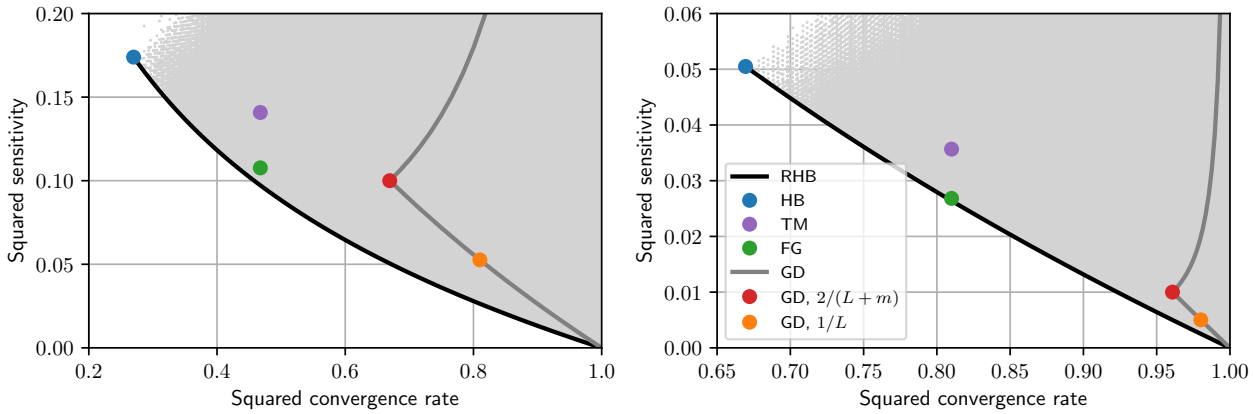

Figure 2: Plot of sensitivity squared vs. rate squared for algorithms applied to the function class $Q_{1,10}$ (left panel) and $Q_{1,100}$ (right panel), found using Theorem 3.2. Each point in the point cloud corresponds to an algorithm (α, β, η). The Pareto-optimal front coincides with our proposed Robust Heavy Ball method (RHB, Theorem 3.4), tuned using $r \in \left[\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}, 1\right]$ to mediate the trade-off.

In Fig. 2, each algorithm (α, β, η) corresponds to a single gray dot The RHB curve shows each possible tuning as we vary the parameter r. We observe that RHB perfectly traces out the boundary of the point cloud, which represents the Pareto-optimal algorithms. In other words, for a fixed convergence rate r, RHB with parameter r achieves this rate and is also as robust as possible to additive gradient noise.

Fig. 2 reveals that GD with $0 < \alpha < \frac{2}{L+m}$ is outperformed by RHB on the function class $Q_{m,L}$. We also plot the performance of GD for $\alpha > \frac{2}{L+m}$, which is even worse as this leads to slower convergence *and* increased sensitivity. Fig. 2 also reveals that FG is strictly suboptimal compared to RHB based on the analysis in Theorem 4.5, although the optimality gap appears to shrink as L/m gets larger.

Smooth strongly convex functions ($F_{m,L}$). We show our brute-force search for the class $F_{m,L}$ in Fig. 3 for $F_{1,10}$ and $F_{1,100}$. For this figure, we used the same sampling approach as in Fig. 2, with $200 \times 51 \times 50$ samples. When applying Theorem 4.5, we used a lifting dimension $\ell = 1$ to compute the rate and $\ell = 6$ to compute the sensitivity. For more details on these choices, see Section 5.4.

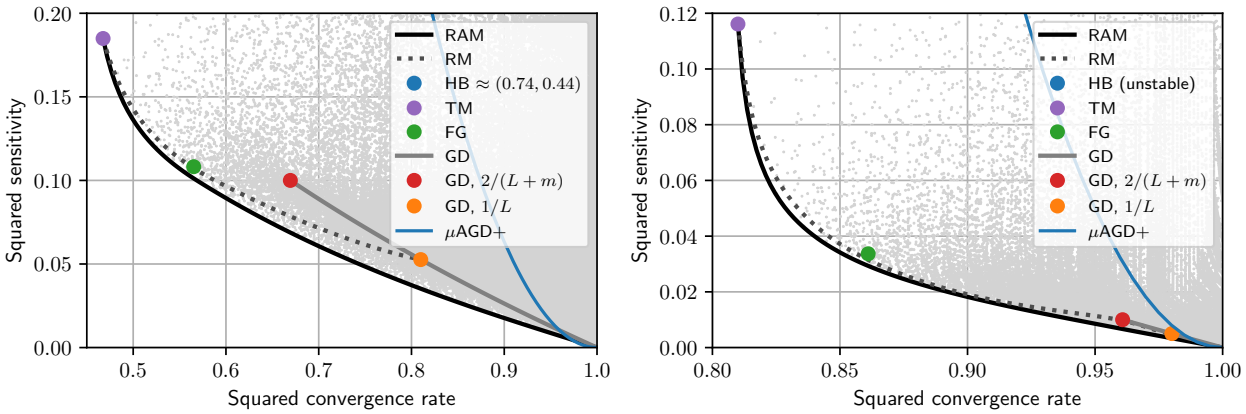

Figure 3: Plot of squared sensitivity vs. squared rate for algorithms applied to the function class $F_{1,10}$ (left panel) and $F_{1,100}$ (right panel), found using Theorem 4.5. Each point in the point cloud corresponds to an algorithm (α, β, η). RM and GD are strictly suboptimal, and the Pareto-optimal front closely matches RAM (Theorem 4.7), tuned using $r \in \left[1 - \sqrt{\frac{m}{L}}, 1\right]$ to mediate the trade-off. For reference, the bound (6) on μAGD+ from [14] is also shown.

The Robust Momentum (RM) method from [15] interpolates between TM and GD with $\alpha = \frac{1}{L}$ and does trade off convergence rate and sensitivity, but based on our analysis in Theorem 4.5 it is strictly outperformed by our proposed RAM. The gap in performance between RM and RAM appears to shrink as L/m gets larger.

While Fig. 3 indicates that RAM effectively trades off rate and robustness, it is strictly suboptimal[7]. Suboptimality becomes most apparent when L/m is small and r is close to 1. For example, consider RAM with the parameter choice $r = 0.9$, $m = 1$, and $L = 2$, which corresponds to $(\alpha, \beta, \eta) = (0.019, 0.66, -3.631579)$. Solving the LMIs in Theorem 4.5 yields the rate 0.9000 and sensitivity 0.22057. However, if we change η and use the tuning $(\alpha, \beta, \eta) = (0.019, 0.66, 0.00)$ instead, we obtain the same rate with the strictly smaller sensitivity 0.1676. Larger optimality gaps can be found by making L/m even closer to 1, although such cases are not practical.

Remark 7. *The left panel of Fig. 3 ($L/m = 10$) shows a denser point cloud than the right panel ($L/m = 100$), despite using the same number of sample points. This occurs because the right panel*

[7]Suboptimality is difficult to verify for the function class $F_{m,L}$ since the results depend on the lifting dimension ℓ. However, we performed extensive numerical computations to ensure that ℓ is sufficiently large; see Section 5.4.

spans a larger range of γ values (the vertical axis is truncated), indicating that desirable algorithm tunings become harder to find by random sampling as L/m increases.

5.2 Justifying the three-parameter algorithm family

A natural question to ask is whether something as general as our three-parameter family (5) is needed to achieve the performance of our designs. Several recent works have restricted their attention to optimizing algorithms with two parameters (α, β) in either Nesterov's FG or Polyak's HB form [5, 28, 43]. From our results in Sections 3.2 and 5.1, the HB form is sufficient for the class $Q_{m,L}$. However, neither the HB or the FG forms are sufficient for $F_{m,L}$. While some algorithms in these restricted classes achieve acceleration, they are incapable of obtaining the same trade-off between convergence rate and sensitivity as a properly-tuned three-parameter method, as illustrated in Fig. 4 (left panel). Indeed, even when there is no noise, no algorithm in the FG or HB families achieves the optimal convergence rate for the function class $F_{m,L}$, which is attained by Van Scoy et al.'s Triple Momentum (TM) method [55].

Alternatively, we could ask whether using more than three parameters could lead to further improvement. As explained in Section 2.2, any algorithm with $n = 2$ states can be represented by three parameters. In general, we would need $2n - 1$ parameters to represent an algorithm with n states. In principle, our methodology of Section 1.3 can still be applied, but the associated semidefinite programs become substantially more difficult to solve and we were unable to find better designs.

An alternative approach, presented in [42], used a convex synthesis procedure and bilinear matrix inequalities to numerically construct algorithms that trade off convergence rate and sensitivity. As shown in Fig. 4 (right panel), these synthesized algorithms do not outperform RAM, despite using up to $n = 6$ states.

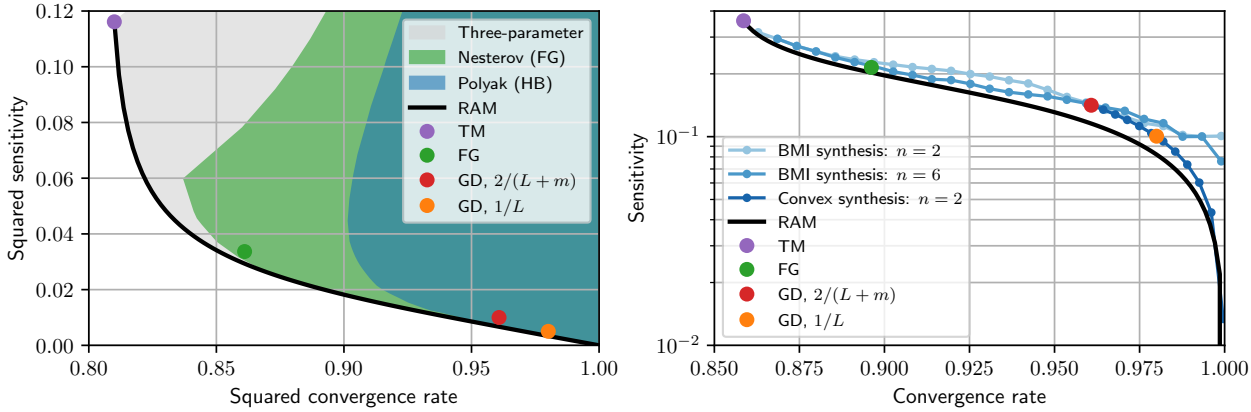

Figure 4: Left: Regions of the sensitivity vs. rate trade-off space for $F_{1,100}$ covered by the three-parameter family (α, β, η), the Nesterov (Fast Gradient) family (α, β, β), and the Polyak (Heavy Ball) family $(\alpha, \beta, 0)$. The FG and HB families are not expressive enough to capture the whole trade-off space. **Right:** Comparison of RAM with the numerically synthesized algorithms (using a state dimension up to $n = 6$) from [42] for $F_{1,50}$. RAM outperforms despite using only two states of memory. We plot the log of the normalized sensitivity vs. the rate to match [42, Fig. 6].

5.3 Simulation of a worst-case test function

We simulated various algorithms on Nesterov's lower-bound function, which is a quadratic with a tridiagonal Hessian [44, §2.1.4]. We used $d = 100$ with $m = 1$ and $L = 10$ and initialized each algorithm at zero. The results are reported in Fig. 5. We tested both a *low noise* ($\sigma = 10^{-5}$, left column) and a *higher noise* ($\sigma = 10^{-2}$, right column) regime. We recorded the mean and standard deviation of the error across 100 trials for each algorithm (the trials differ only in the noise realization).

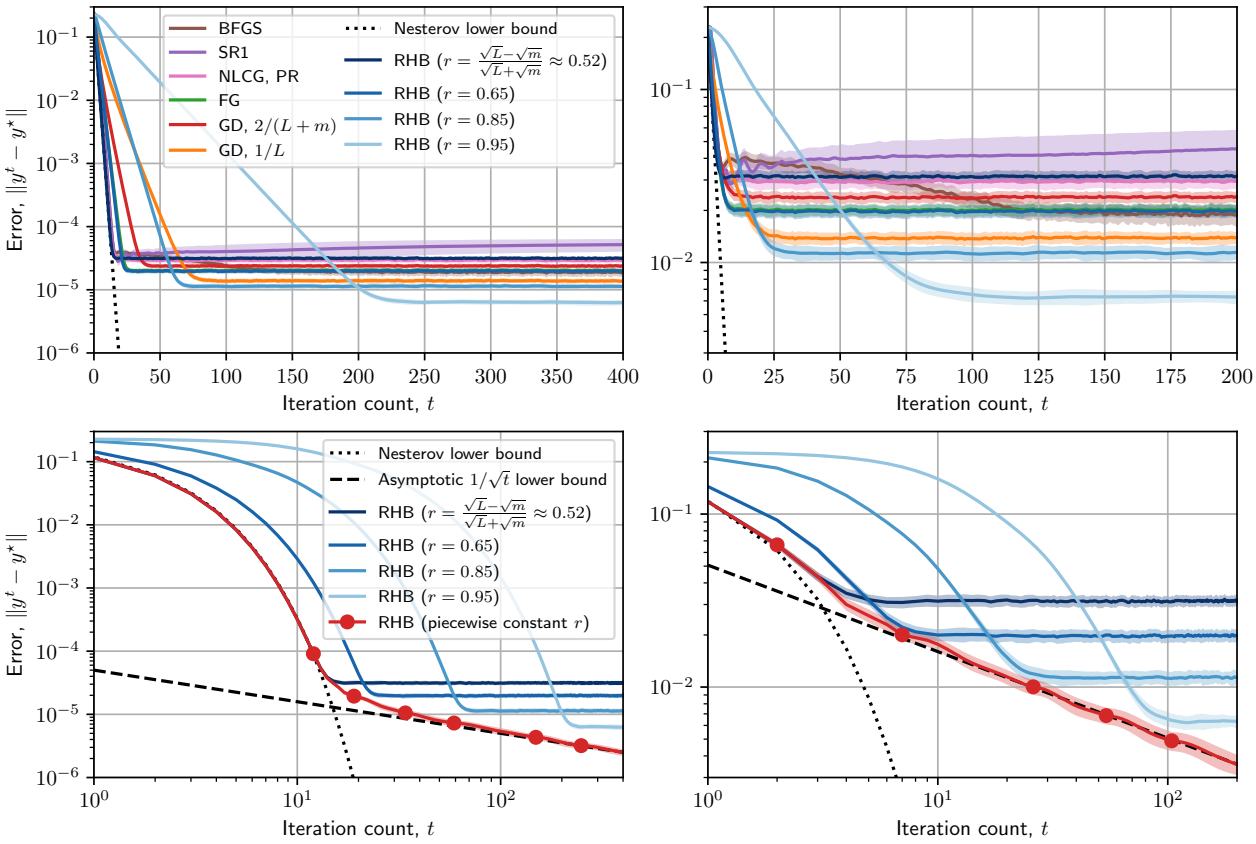

Figure 5: Simulations of various algorithms with low noise ($\sigma = 10^{-5}$, left column) and higher noise ($\sigma = 10^{-2}$, right column). Each algorithm was simulated on Nesterov's lower-bound quadratic function, with $m = 1$, $L = 10$, and dimension $d = 100$. Shaded regions indicate ± 1 standard deviations about the mean across 100 trials (different noise realizations). Different tunings of our proposed Robust Heavy Ball (RHB) from Theorem 3.4 effectively trade off convergence rate and steady-state error (sensitivity to noise). Bottom row: the red curve shows RHB with (hand-tuned) piecewise constant r, where the red dots indicate switch points.

Fig. 5 shows that RHB from Theorem 3.4 trades off convergence rate (the initial decreasing slope) with sensitivity to noise (the steady-state value), and compares favorably to other methods. The other methods we tested (first row of Fig. 5) are generally suboptimal compared to RHB, in the sense that there is some choice of tuning parameter r such that RHB is both faster and has smaller steady-state error.

In addition to GD and FG, we tested Nonlinear Conjugate Gradient (NLCG) with the Polak-Ribière

(PR) update scheme.[8] NLCG performs similarly to RHB with the most aggressive tuning, which is equivalent to the Heavy Ball method. We also tested the popular quasi-Newton methods [46] Broyden–Fletcher–Goldfarb–Shanno (BFGS) and Symmetric Rank-One (SR1), which performed strictly worse than RHB.[9]

In the second row of Fig. 5, we use the same settings as the first row, except iterations are plotted on a log scale as in Fig. 1. We also show a hand-tuned version of RHB with piecewise constant r.[10] Whenever r is changed, we re-initialize the algorithm via $x^{t-1} = x^t$. In the transient phase, the hand-tuned RHB matches Nesterov's lower bound (11). In the stationary phase, it matches the asymptotic lower bound (slope of $-1/2$) described in Section 1.2.[11]

5.4 Computational considerations

We used Julia v.1.12.1 [7] for all computation, and our code is available at `https://github.com/QCGroup/optalg`. Applying Theorem 4.5 required picking ℓ. Using larger ℓ could reduce conservatism, but at the cost of larger LMIs. To this end, we performed a pilot study using the arbitrary-precision solver Hypatia [13]. In Table 2, we apply Theorem 4.5 and report the performance of FG with standard tuning (see Table 1), with 30 significant digits[12]. The rate did not improve beyond $\ell = 1$ but the sensitivity bound continued to improve.

To generate Fig. 3, we solved the LMIs from Theorem 4.5 using JuMP v.1.29.2 [40] with the Mosek solver v.11.0.1 [3] and default settings. We used $\ell = 1$ for the rate, with a bisection search tolerance of 10^{-6}, and $\ell = 6$ for the sensitivity. With these choices, finding the rate and sensitivity of a given three-parameter algorithm with 5 significant digits of precision each took about 100 ms on a conventional laptop.

6 Concluding remarks

For $Q_{m,L}$ and $F_{m,L}$, we provided (i) efficient methods for computing the convergence rate and noise sensitivity for a broad class of first-order methods, and (ii) first-order algorithms designs, each with a single tunable parameter that directly trades off convergence rate and sensitivity to additive gradient noise.

An interesting future direction is exploring adaptive versions of these algorithms, where the parameter r is varied over time. We showed in Fig. 5 that a hand-tuned piecewise constant version of

[8] We also tested other popular NLCG update schemes such as: Fletcher–Reeves, Hestenes–Stiefel, and Dai–Yuan; all produced similar trajectories to PR.

[9] Both NLCG and BFGS use line search; given a current point $y \in \mathbb{R}^d$ and search direction $s \in \mathbb{R}^d$, they search for $\alpha \in \mathbb{R}$ that minimizes $f(y+\alpha s)$. In practice, *inexact* line searches are performed at each timestep with a stopping criterion such as the Wolfe conditions. To show these algorithms in the most charitable light, we used exact line searches but substituted the noisy gradient oracle. Specifically, with $f(y) = \frac{1}{2}(y-y^\star)^\mathsf{T} Q(y-y^\star)$, the optimal stepsize is $\alpha^\star = -(s^\mathsf{T}\nabla f(y))/(s^\mathsf{T} Qs)$. We used this formula, but replaced $\nabla f(y)$ by the noisy $\nabla f(y) + w$ (exact knowledge of Q but not $y^\star$).

[10] We use a hand-tuned schedule to illustrate our results; more systematic scheduling methods have been proposed, such as the RESTART+SLOWDOWN method in [14].

[11] The lower bound is $1/t$ for the squared error, hence $1/\sqrt{t}$ for the error, which appears as a line of slope $-1/2$ on the log-log scale of Fig. 5, bottom row.

[12] For each case, we computed optimal primal and dual solutions, verified that each was strictly feasible, and ensured the duality gap was less than 10^{-30}.

Table 2: Performance bounds for FG with standard tuning (see Table 1) on $F_{1,100}$ found by solving the LMIs from Theorem 4.5 using different lifting dimensions ℓ and arbitrary-precision arithmetic. All digits shown are significant, and digits that represent conservatism due to choice of ℓ are in red. The last digit is rounded up so all numbers shown represent valid upper bounds.

ℓ	Rate upper bound	Sensitivity upper bound
1	0.92793311096382285055040597129 5	0.20076555494166540966517022265 3
2	0.92793311096382285055040597129 5	0.18590868937237265544941519972 7
3	0.92793311096382285055040597129 5	0.18372874155768134467047518447 6
4	0.92793311096382285055040597129 5	0.18351362872781827645186170585 6
5	0.92793311096382285055040597129 5	0.18349324602795443380975483354 4
6	0.92793311096382285055040597129 5	0.18349130905507473094591977809 1
7	0.92793311096382285055040597129 5	0.18349112475553019786987381552 3
8	0.92793311096382285055040597129 5	0.18349110721545996883777768926 4
9	0.92793311096382285055040597129 5	0.18349110554607966257673462480 9
10	0.92793311096382285055040597129 5	0.18349110538719512206717817909 6

RHB can match both Nesterov's lower bound and the gradient lower bound, so more sophisticated adaptive schemes such as those described in Section 1.2 might also work. One could also adjust parameters continually, but proving the convergence of adaptive algorithms is generally more challenging. For example, the well-known ADMM algorithm is often tuned adaptively to improve transient performance, even when convergence guarantees only hold for fixed parameters [10, §3.4.1.]. Nevertheless, LMI-based approaches have been successfully used to prove convergence of algorithms with time-varying parameters [26, 31].

Another interesting open question is whether our analysis is tight. For $F_{m,L}$ (Theorem 4.5), our bounds depend on ℓ. It is unknown (i) how large ℓ needs to be in order to obtain the tightest possible bounds on convergence rate and sensitivity, and (ii) whether Theorem 4.5 always produces tight bounds as $\ell \to \infty$.

A Detailed proofs

A.1 Proof of Theorem 2.1 (3-parameter family)

The three-parameter family (5) is of the general form (7) with $n = 2$;

$$\hat{\xi}^t = \begin{bmatrix} x^t \\ x^{t-1} \end{bmatrix}, \qquad \hat{A} = \begin{bmatrix} 1+\beta & -\beta \\ 1 & 0 \end{bmatrix}, \qquad \hat{B} = \begin{bmatrix} -\alpha \\ 0 \end{bmatrix}, \qquad \hat{C} = \begin{bmatrix} 1+\eta & -\eta \end{bmatrix}. \tag{27}$$

Let (A, B, C) be a general algorithm of the form (7) satisfying Assumption 1 with $n = 2$. We will construct a transformation matrix T, as introduced in Remark 2, that transforms (A, B, C) to the form of (27).

Let the eigenvalues of A be $\{1, \beta\}$ with corresponding eigenvectors $v_1, v_2 \in \mathbb{R}^2$. First transform using $T_1 = \begin{bmatrix} v_1 & v_2 \end{bmatrix}^{-1}$, which has the effect of diagonalizing A, and obtain $(A, B, C) \mapsto \left(\left[\begin{smallmatrix} 1 & 0 \\ 0 & \beta \end{smallmatrix}\right], \left[\begin{smallmatrix} b_1 \\ b_2 \end{smallmatrix}\right], \left[\begin{smallmatrix} c_1 & c_2 \end{smallmatrix}\right] \right)$. Then, transform using $T_2 = \frac{c_1}{b_2} \left[\begin{smallmatrix} b_2 & -\beta b_1 \\ b_2 & -b_1 \end{smallmatrix}\right]$ and obtain

$\left(\left[\begin{smallmatrix}1+\beta & -\beta\\ 1 & 0\end{smallmatrix}\right], \left[\begin{smallmatrix}(1-\beta)b_1c_1\\ 0\end{smallmatrix}\right], \left[\begin{smallmatrix}1+\frac{\beta b_1c_1+b_2c_2}{(1-\beta)b_1c_1} & -\frac{\beta b_1c_1+b_2c_2}{(1-\beta)b_1c_1}\end{smallmatrix}\right]\right)$. Setting $\alpha = -(1-\beta)b_1c_1$ and $\eta = \frac{\beta b_1c_1+b_2c_2}{(1-\beta)b_1c_1}$, the transformed system matches (27). Thus, applying the transformation $T = T_2T_1$ transforms a generic (A, B, C) to the form (5) and if we initialize the new system with state $\hat{\xi}_0 = T\xi_0$, where ξ_0 is the initial state of (A, B, C), both systems will produce the same sequence of iterates $(y^0, y^1, \dots)$. ■

A.2 Proof of Theorem 3.1 ($Q_{m,L}$ analysis)

Consider a particular $f \in Q_{m,L}$. We can write $\nabla f(y) = (y - y^\star)Q$ for some $Q \in \mathbb{R}^{d\times d}$ satisfying $Q = Q^\mathsf{T} \succ 0$ (recall that the iterates are row vectors, so $y \in \mathbb{R}^{1\times d}$). The dynamics (7) become

$$\xi^{t+1} - \xi^\star = A(\xi^t - \xi^\star) + BC(\xi^t - \xi^\star)Q + Bw^t \quad \text{and} \quad y^t - y^\star = C(\xi^t - \xi^\star). \tag{28}$$

If we diagonalize Q as $V\Lambda V^\mathsf{T}$ with $V = \begin{bmatrix}v_1 & \dots & v_d\end{bmatrix}$ and $\Lambda = \operatorname{diag}(q_1, \dots, q_d)$, we can split the dynamics into d decoupled systems by multiplying on the right by v_i and letting $\hat{\xi}_i^t := (\xi^t - \xi^\star)v_i$ and similarly for $\hat{y}_i^t$ and $\hat{w}_i^t$, obtaining

$$\hat{\xi}_i^{t+1} = (A + q_iBC)\hat{\xi}_i^t + B\hat{w}_i^t \qquad \text{and} \qquad \hat{y}_i^t = C\hat{\xi}_i^t. \tag{29}$$

We now have $\hat{\xi}_i^t \in \mathbb{R}^{n\times 1}$, $\hat{y}_i^t \in \mathbb{R}$, and $\hat{w}_i^t \in \mathbb{R}$. The rate and sensitivity of (28) can be found by analyzing the simpler system (29). Specifically, the rate for $\mathcal{A}$ applied to f is $\max_{i\in\{1,\dots,d\}} \rho(A + q_iBC)$, so the worst case over $Q_{m,L}$ is the supremum of $\rho(A + qBC)$ over all $q \in [m, L]$ (the set of possible eigenvalues of Q), as required.

For sensitivity, define $X_i^t := \mathbb{E}\left[(\hat{\xi}_i^t)(\hat{\xi}_i^t)^\mathsf{T}\right] \in \mathbb{R}^{n\times n}$, $W_i^t := \mathbb{E}\left[(\hat{w}_i^t)^2\right] \in \mathbb{R}$, and $Y_i^t := \mathbb{E}\left[(\hat{y}_i^t)^2\right] \in \mathbb{R}$. By assumption, $\sum_{i=1}^d W_i^t \le \sigma^2$ for all t and $W_i^t \ge 0$ for all i, t. The worst-case sensitivity for $\mathcal{A}$ applied to f is achieved when all weight is placed on a single index i for all t, so $W_i^t = W_i = \sigma^2$. For this i, we can write the recurrence

$$X_i^{t+1} = (A + q_iBC)X_i^t(A + q_iBC)^\mathsf{T} + \sigma^2BB^\mathsf{T} \quad \text{and} \quad Y_i^t = CX_i^tC^\mathsf{T}. \tag{30}$$

Define the Lyapunov operator $\mathcal{L}_i(Z) := \sum_{k=0}^\infty (A+q_iBC)^k Z\left((A+q_iBC)^\mathsf{T}\right)^k$. This series converges whenever $\rho(A+q_iBC) < 1$ to the (unique) solution X of the Lyapunov equation $(A+q_iBC)X(A+q_iBC)^\mathsf{T} - X + Z = 0$. Since $\textsc{rate}(\mathcal{A}, Q_{m,L}) < 1$ by assumption, $\mathcal{L}_i$ is well-defined for all $q_i \in [m, L]$. Applying $\mathcal{L}_i$ above, we obtain

$$\limsup_{T\to\infty} \mathbb{E}_{w\sim\mathbb{P}} \frac{1}{T\sigma^2} \sum_{t=0}^{T-1} \|y^t - y^\star\|^2 \le \max_{1\le i\le d} \frac{1}{\sigma^2} C\mathcal{L}_i(\sigma^2BB^\mathsf{T})C^\mathsf{T} = \max_{1\le i\le d} C\mathcal{L}_i(BB^\mathsf{T})C^\mathsf{T},$$

where the final equality is due to linearity of $\mathcal{L}_i$. The worst case over $Q_{m,L}$ is then the supremum of $\sqrt{C\mathcal{L}(BB^\mathsf{T})C^\mathsf{T}}$ over all possible eigenvalues $q \in [m, L]$ of Q. Define the adjoint Lyapunov operator $\mathcal{L}^*(Z) := \sum_{k=0}^\infty \left((A + qBC)^\mathsf{T}\right)^k Z(A + qBC)^k$ as the unique solution Y to the Lyapunov equation $(A + qBC)^\mathsf{T}Y(A + qBC) - Y + Z = 0$. Applying Lagrangian duality, we have $\sqrt{C\mathcal{L}(BB^\mathsf{T})C^\mathsf{T}} = \sqrt{B^\mathsf{T}\mathcal{L}^*(C^\mathsf{T}C)B}$. ■

A.3 Proof of Theorem 3.2 (reduced analysis for $Q_{m,L}$)

We first prove the rate expression (12). Substituting the algorithm form (27) into Theorem 3.1,

$$\textsc{rate}(\mathcal{A}, Q_{m,L}) = \sup_{q \in [m,L]} \phi(q) := \rho\left(\begin{bmatrix} \beta + 1 - \alpha(\eta+1)q & -\beta + \alpha\eta q \\ 1 & 0 \end{bmatrix}\right). \tag{31}$$

We will prove that $\phi(q)$ defined above is quasiconvex [11, §3.4]. The characteristic polynomial of the matrix in (31) is $\chi(z) = z^2 + (\alpha(\eta+1)q - \beta - 1)z + (\beta - \alpha\eta q)$. Given a polynomial with real coefficients, a necessary and sufficient condition for its roots to lie inside the unit circle is given by the Jury test [24, §4.5]. For the quadratic $z^2 + a_1 z + a_0$, the Jury test reduces to $1 + a_1 + a_0 > 0$, $1 - a_1 + a_0 > 0$, and $-1 < a_0 < 1$. Applying the Jury test to $\chi(rz)$, we find that $\phi(q) < r$ if and only if

$$\begin{aligned} (1-r)(\beta - r) + \alpha(\eta r - \eta + r)q > 0, \quad r^2 + \beta - \alpha\eta q > 0, \\ (1+r)(\beta + r) - \alpha(\eta r + \eta + r)q > 0, \quad r^2 - \beta + \alpha\eta q > 0. \end{aligned} \tag{32}$$

The inequalities (32) are linear in q, so the sublevel sets $\{q \mid \phi(q) < r\}$ are open intervals. Therefore, ϕ is quasiconvex and attains its supremum over $q \in [m, L]$ at one of the endpoints, $q = m$ or $q = L$. The explicit formula for $\phi(q)$ can be found by applying the quadratic formula to find the roots of $\chi(z)$. In (12), Δ is the discriminant of $\chi(z)$ and the two cases correspond to whether the roots are real or complex.

We next prove the expression for the sensitivity in (13). Substituting the algorithm form (27) into Theorem 3.1, we can explicitly solve the linear equation for P_q in (13) to obtain $h(q) = B^{\mathsf{T}} P_q B$, where $h(q)$ is defined in (13). When $r = 1$, the Jury conditions (32) reduce to:

$$\alpha q > 0, \tag{33a}$$
$$2\beta + 2 - \alpha(2\eta+1)q > 0, \tag{33b}$$
$$1 + \beta - \alpha\eta q > 0, \tag{33c}$$
$$1 - \beta + \alpha\eta q > 0. \tag{33d}$$

When the rate is strictly less than one, $h(q)$ is positive and convex. To see why, evaluate $h(q)$ and $h''(q)$. After routine algebraic manipulations, we obtain:

$$h(q) = \frac{\alpha^2(2\eta+1)^2}{2(1-\beta+\alpha\eta q)(2\beta+2-\alpha(2\eta+1)q)} + \frac{\alpha^2}{2\alpha q(\alpha\eta q - \beta + 1)}, \tag{34a}$$

$$\begin{aligned} h''(q) = {} & \frac{\alpha^4(2\eta+1)^2\Big(3(2\alpha\eta(2\eta+1)q - 4\beta\eta - \beta + 1)^2 + (4\eta - \beta + 1)^2\Big)}{4(1-\beta+\alpha\eta q)^3(2\beta+2-\alpha(2\eta+1)q)^3} \\ & + \frac{\alpha q\Big(3\big(2\alpha\eta q + 1 - \beta\big)^2 + \big(1-\beta\big)^2\Big)}{4q^4(\alpha\eta q - \beta + 1)^3}. \end{aligned} \tag{34b}$$

In the form (34), it is clear that whenever the rate is strictly less than one (i.e., when (33) holds) we have $h(q) > 0$ and $h''(q) > 0$. Therefore, the quantity under the square root in (13) is always positive and $h(q)$ is convex, so it attains its supremum over $q \in [m, L]$ at one of the endpoints, $q = m$ or $q = L$. ∎

A.4 Proof of Theorem 4.5 ($F_{m,L}$ analysis)

Consider a trajectory of the dynamics (ξ^t, u^t, y^t, w^t) (7) with $w^t = 0$. Form the augmented vectors and state as described in Section 4.4. Multiply the LMIs (23a) and (23c) on the right and left by $\mathrm{col}(\boldsymbol{\xi}^t, u^t) \in \mathbb{R}^{(n+2\ell)\times d}$ and its transpose, respectively, and take the trace. Also, take the inner product of (23b) and (23d) with $\boldsymbol{f}^t$, which preserves the inequality because $\boldsymbol{f}^t$ is elementwise nonnegative. The resulting inequalities are:

$$\operatorname{tr}(\boldsymbol{\xi}^{t+1})^\mathsf{T} P \boldsymbol{\xi}^{t+1} - r^2 \operatorname{tr}(\boldsymbol{\xi}^t)^\mathsf{T} P \boldsymbol{\xi}^t + \operatorname{tr} \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix}^\mathsf{T} \Pi(\Lambda_1) \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix} \leq 0, \tag{35a}$$

$$p^\mathsf{T}(Z_+ - r^2 Z)\boldsymbol{f}^t + \pi(\Lambda_1)^\mathsf{T} \boldsymbol{f}^t \leq 0, \tag{35b}$$

$$\|\xi^t - \xi^\star\|^2 - \operatorname{tr}(\boldsymbol{\xi}^t)^\mathsf{T} P \boldsymbol{\xi}^t + \operatorname{tr} \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix}^\mathsf{T} \Pi(\Lambda_1) \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix} \leq 0, \tag{35c}$$

$$-p^\mathsf{T} Z \boldsymbol{f}^t + \pi(\Lambda_2)^\mathsf{T} \boldsymbol{f}^t \leq 0. \tag{35d}$$

Summing (35a)+(35b) and (35c)+(35d) and applying (20), we recover the lower bound and decrease properties in Theorem 4.1 and the rate bound result follows.

For the second part of the proof, we do not restrict $w^t = 0$, and perform similar operations to the inequalities (24) as in the first part to obtain the inequalities

$$\operatorname{tr}(\boldsymbol{\xi}^{t+1} - \boldsymbol{H}w^t)^\mathsf{T} P (\boldsymbol{\xi}^{t+1} - \boldsymbol{H}w^t) - \operatorname{tr}(\boldsymbol{\xi}^t)^\mathsf{T} P \boldsymbol{\xi}^t + \operatorname{tr} \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix}^\mathsf{T} \Pi(\Lambda_1) \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix} + \|\tilde{y}^t\|^2 \leq 0, \tag{36a}$$

$$p^\mathsf{T}(Z_+ - Z)\boldsymbol{f}^t + \pi(\Lambda_1)^\mathsf{T} \boldsymbol{f}^t \leq 0, \tag{36b}$$

$$-\operatorname{tr}(\boldsymbol{\xi}^t)^\mathsf{T} P \boldsymbol{\xi}^t + \operatorname{tr} \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix}^\mathsf{T} \Pi(\Lambda_2) \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{u}^t \end{bmatrix} \leq 0, \tag{36c}$$

$$-p^\mathsf{T} Z \boldsymbol{f}^t + \pi(\Lambda_2)^\mathsf{T} \boldsymbol{f}^t \leq 0, \tag{36d}$$

where $\tilde{y}^t = y^t - y^\star$. Summing (36a)+(36b) and (36c)+(36d), applying (20), and using the definition of the augmented state (16) and the Lyapunov function (22),

$$V(\boldsymbol{x}^{t+1}) - V(\boldsymbol{x}^t) + \|\tilde{y}^t\|^2 - 2\operatorname{tr}(\boldsymbol{A}\boldsymbol{\xi}^t + \boldsymbol{B}u^t)^\mathsf{T} P \boldsymbol{H} w^t - \operatorname{tr}(w^t)^\mathsf{T} \boldsymbol{H}^\mathsf{T} P \boldsymbol{H} w^t \leq 0$$

and $V(\boldsymbol{x}^t) \geq 0$. Taking the expectation of both inequalities, the term $-2(\boldsymbol{A}\boldsymbol{\xi}^t + \boldsymbol{B}u^t)^\mathsf{T} P \boldsymbol{H} w^t$ in the first inequality vanishes because w^t is zero-mean and is independent of $\boldsymbol{\xi}^t$ and u^t, which only depend on $w^{t-1}, w^{t-2}, \ldots$. Also, since w^t has expected squared norm bounded by σ^2, we have $\operatorname{tr} \mathbb{E}\big((w^t)^\mathsf{T} \boldsymbol{H}^\mathsf{T} P \boldsymbol{H} w^t\big) \leq \sigma^2(\boldsymbol{H}^\mathsf{T} P \boldsymbol{H})$. Thus, the previous inequalities imply the decrease and lower bound conditions in Theorem 4.2 with $\gamma^2 = \sigma^2(\boldsymbol{H}^\mathsf{T} P \boldsymbol{H})$, which implies the bound on the sensitivity. ∎

A.5 Proof of Theorem 4.6 (GD analysis for $F_{m,L}$)

Setting $\ell = 0$ in Theorem 4.5 and applying Remark 5 with the GD parameters from (7) ($A = C = 1$ and $B = -\alpha$), the LMIs (23)–(24) with stepsize $\alpha = \frac{1}{m}(1-r)$ are satisfied with $P_1 = P_2 = \frac{1}{1-r^2}$ and $\lambda_1 = \lambda_2 = \frac{r}{m(L-m)(r+1)}$ for all $\frac{L-m}{L+m} \leq r < 1$. Therefore, $\textsc{rate}(\mathcal{A}, F_{m,L}) \leq r$ and

$\textsc{sens}(\mathcal{A}, F_{m,L}) \leq \alpha\sqrt{P_2} = \frac{1}{m}\sqrt{\frac{1-r}{1+r}}$. Since these upper bounds match the exact values found for $Q_{m,L}$ in Theorem 3.3, we conclude the upper bounds are tight, with the worst-case function being the quadratic $f(y) = \frac{m}{2}\|y\|^2$. ∎

A.6 Proof of Theorem 4.7 (RAM analysis for $F_{m,L}$)

To prove that RAM converges with rate r when there is no noise, we provide a feasible solution to the LMI (23) with $\ell = 1$. Our solution has the same structure as the weighted off-by-one IQC formulation in (25), where the positive definite matrix $Q \succ 0$ is given by

$$Q = \frac{m}{(3-r)(1-r)^2(1+r)^3(m - L\,(1-r)^2)} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix},$$

with:
$$\begin{aligned}
q_{11} &= r\left(2m^2(1-r) + 2Lm\,(4 + r - 2r^2 + r^3) - L^2(1-r^2)^2\right), \\
q_{12} &= r\left(-2m^2(1-r) - 2Lm\,(1+r)^2 + L^2(4-r)(1-r^2)^2\right), \\
q_{13} &= (3-r)(1-r^2)\big(-m\,(1+r^2) + L\,(1 + r - 2r^2 - r^3 + r^4)\big), \\
q_{22} &= r\left(2m^2(1-r) - 2Lm\,(2 - 3r - 4r^2 + r^3) + L^2(2 - 4r + r^2)(1-r^2)^2\right), \\
q_{23} &= r\,(3-r)(1-r^2)\big(m\,(-1 + 2r + r^2) - L\,(-1 + r - r^3 + r^4)\big), \\
q_{33} &= r\,(3-r)^2(1-r^2)^2.
\end{aligned}$$

Using Mathematica [59], we verify that, for all $0 < m \leq L$ with $1 - \sqrt{\frac{m}{L}} \leq r < 1$, this is a feasible solution to a modified version of the LMI (23) in which the term $\boldsymbol{X}_r^\mathsf{T}\boldsymbol{X}_r$ is replaced by $\boldsymbol{X}_r^\mathsf{T}T^\mathsf{T}QT\boldsymbol{X}_r$, where $T = \left[\begin{smallmatrix} A & B \\ -LC & 1 \end{smallmatrix}\right]$. Since $T^\mathsf{T}QT \succ 0$, we have $\boldsymbol{X}_r^\mathsf{T}T^\mathsf{T}QT\boldsymbol{X}_r \succeq c\,\boldsymbol{X}_r^\mathsf{T}\boldsymbol{X}_r$, where $c > 0$ is the minimum eigenvalue of $T^\mathsf{T}QT$. Therefore, scaling the solution by $1/c$ yields a feasible solution to the original LMI (23). Theorem 4.5 then implies that RAM has convergence rate at most r. Since $\rho(A + mBC) = r$, the bound is achieved by $f(y) = \frac{m}{2}\|y\|^2$, so the convergence rate is *exactly* r. ∎

A.7 Proof of Theorem 5.1 (parameter restriction)

The Jury criterion for stability ($r < 1$) is given in (33). Combining (33b) $+ 2 \cdot$ (33d) with (33a), we obtain: $0 < \alpha q < 4$. For this to hold for all $q \in [m, L]$, we must have $0 < \alpha < \frac{4}{L}$. Combining (33c) and (33d), we obtain $-1 + \alpha\eta q < \beta < 1 + \alpha\eta q$, which holds for all $q \in [m, L]$. When $\alpha\eta \geq 0$, the β range reduces to $-1 + L(\alpha\eta) < \beta < 1 + m(\alpha\eta)$ and thus, $\alpha\eta < \frac{2}{L-m}$. When $\alpha\eta < 0$, we instead obtain $-1 + m(\alpha\eta) < \beta < 1 + L(\alpha\eta)$ and $\frac{-2}{L-m} < \alpha\eta$. Although these bounds are derived for $Q_{m,L}$, the nestedness property $Q_{m,L} \subseteq F_{m,L}$ implies that these necessary conditions on (α, β, η) also hold for $F_{m,L}$. ∎

References

[1] Z. Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *ACM SIGACT Symposium on Theory of Computing*, pages 1200–1205, 2017.

[2] P. J. Antsaklis and A. N. Michel. *Linear systems*. Springer Science & Business Media, 2006.

[3] M. ApS. *The MOSEK optimization suite 9.2.49*, 2021.

[4] N. S. Aybat, A. Fallah, M. Gürbüzbalaban, and A. Ozdaglar. A universally optimal multistage accelerated stochastic gradient method. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[5] N. S. Aybat, A. Fallah, M. Gurbuzbalaban, and A. Ozdaglar. Robust accelerated gradient methods for smooth strongly convex functions. *SIAM Journal on Optimization*, 30(1):717–751, 2020.

[6] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. *IEEE 55th Annu. Symp. on Found. of Computer Science*, 2014.

[7] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[8] B. Birand, H. Wang, K. Bergman, and G. Zussman. Measurements-based power control - a cross-layered framework. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2013*, 2013.

[9] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[10] S. Boyd, N. Parikh, and E. Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[11] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[12] J. Chee and P. Toulis. Convergence diagnostics for stochastic gradient descent with constant learning rate. In *International Conference on Artificial Intelligence and Statistics*, pages 1476–1485, 2018.

[13] C. Coey, L. Kapelevich, and J. P. Vielma. Performance enhancements for a generic conic interior point algorithm. *Mathematical Programming Computation*, 15:53–101, 2023.

[14] M. Cohen, J. Diakonikolas, and L. Orecchia. On acceleration with noise-corrupted gradients. In *International Conference on Machine Learning*, pages 1019–1028, 2018.

[15] S. Cyrus, B. Hu, B. Van Scoy, and L. Lessard. A robust accelerated optimization algorithm for strongly convex functions. In *American Control Conference*, pages 1376–1381, 2018.

[16] 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*, 11:1185–1199, 2017.

[17] E. De Klerk, F. Glineur, and A. B. Taylor. Worst-case convergence analysis of inexact gradient and newton methods through semidefinite programming performance estimation. *SIAM Journal on Optimization*, 30(3):2053–2082, 2020.

[18] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[19] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods with inexact oracle: the strongly convex case. Core discussion papers; 2013/16, Université Catholique de Louvain, 2013.

[20] O. Devolder, F. Glineur, and Y. Nesterov. Intermediate gradient methods for smooth convex problems with inexact oracle. Core discussion papers; 2013/17, Université Catholique de Louvain, 2013.

[21] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146:37–75, 2014.

[22] Y. Drori and A. B. Taylor. On the oracle complexity of smooth strongly convex minimization. *Journal of Complexity*, 68(C), 2022.

[23] Y. Drori and M. Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145:451–482, 2014.

[24] M. S. Fadali and A. Visioli. *Digital control engineering: analysis and design*. Academic Press, 2013.

[25] C. Fang, C. J. Li, Z. Lin, and T. Zhang. SPIDER: Near-optimal non-convex optimization via stochastic path integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 687–697, 2018.

[26] M. Fazlyab, A. Ribeiro, M. Morari, and V. M. Preciado. Analysis of optimization algorithms via integral quadratic constraints: Nonstrongly convex problems. *SIAM Journal on Optimization*, 28(3):2654–2689, 2018.

[27] R. Ge, S. M. Kakade, R. Kidambi, and P. Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[28] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson. Global convergence of the Heavy-ball method for convex optimization. In *2015 European Control Conference*, pages 310–315, 2015.

[29] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization I: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.

[30] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, II: Shrinking procedures and optimal algorithms. *SIAM Journal on Optimization*, 23(4):2061–2089, 2013.

[31] B. Hu and L. Lessard. Dissipativity theory for Nesterov's accelerated method. In *International Conference on Machine Learning*, pages 1549–1557, 2017.

[32] B. Hu, P. Seiler, and L. Lessard. Analysis of biased stochastic gradient descent using sequential semidefinite programs. *Mathematical Programming*, 187:383–408, 2020.

[33] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Accelerating stochastic gradient descent for least squares regression. In *Conference On Learning Theory*, volume 75, pages 545–604, 2018.

[34] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[35] A. Kulunchakov and J. Mairal. A generic acceleration framework for stochastic composite optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[36] G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133:365–397, 2012.

[37] L. Lessard. The analysis of optimization algorithms: A dissipativity approach. *IEEE Control Systems Magazine*, 42(3):58–72, 2022.

[38] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[39] L. Lessard and P. Seiler. Direct synthesis of iterative algorithms with bounds on achievable worst-case convergence rate. In *American Control Conference*, 2020.

[40] M. Lubin, O. Dowson, J. Dias Garcia, J. Huchette, B. Legat, and J. P. Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 2023.

[41] A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830, 1997.

[42] S. Michalowsky, C. Scherer, and C. Ebenbauer. Robust and structure exploiting optimisation algorithms: an integral quadratic constraint approach. *International Journal of Control*, 94(11):2956–2979, 2021.

[43] H. Mohammadi, M. Razaviyayn, and M. R. Jovanović. Robustness of accelerated first-order algorithms for strongly convex optimization problems. *IEEE Transactions on Automatic Control*, 66(6):2480–2495, 2021.

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

[45] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621, 2017.

[46] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

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

[48] E. Ryu, A. B. Taylor, C. Bergeling, and P. Giselsson. Operator splitting performance estimation: Tight contraction factors and optimal parameter selection. *SIAM Journal on Optimization*, 30(3):2251–2271, 2020.

[49] C. Scherer and C. Ebenbauer. Convex synthesis of accelerated gradient algorithms. *SIAM Journal on Control and Optimization*, 59(6):4615–4645, 2021.

[50] A. B. Taylor and F. Bach. Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions. In *Conference on Learning Theory*, volume 99, pages 2934–2992, 2019.

[51] A. B. Taylor and Y. Drori. An optimal gradient method for smooth strongly convex minimization. *Mathematical Programming*, 199:557–594, 2022.

[52] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, 161:307–345, 2017.

[53] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Exact worst-case convergence rates of the proximal gradient method for composite convex minimization. *Journal of Optimization Theory and Applications*, 178(2):455–476, 2018.

[54] A. B. Taylor, B. Van Scoy, and L. Lessard. Lyapunov functions for first-order methods: Tight automated convergence guarantees. In *International Conference on Machine Learning*, pages 4897–4906, 2018.

[55] 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 System Letters*, 2(1):49–54, 2017.

[56] B. Van Scoy and L. Lessard. Absolute stability via lifting and interpolation. In *IEEE Conference on Decision and Control*, pages 6217–6223, 2022.

[57] B. Van Scoy and L. Lessard. A tutorial on a Lyapunov-based approach to the analysis of iterative optimization algorithms. In *IEEE Conference on Decision and Control*, pages 3003–3008, 2023.

[58] C. Wang, X. Chen, A. J. Smola, and E. P. Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[59] Wolfram Research, Inc. Mathematica, Version 12.3.1. Champaign, IL, 2021.

[60] G. Zames and P. Falb. Stability conditions for systems with monotone and slope-restricted nonlinearities. *SIAM Journal on Control*, 6(1):89–108, 1968.

[61] D. Zhou, P. Xu, and Q. Gu. Stochastic nested variance reduction for nonconvex optimization. *Journal of Machine Learning Research*, 21(103):1–63, 2020.