The Analysis of Optimization Algorithms
A dissipativity approach

Laurent Lessard
Department of Mechanical and Industrial Engineering
Northeastern University (l.lessard@northeastern.edu)

Optimization algorithms are ubiquitous across all branches of engineering, computer science, and applied mathematics. Typically, such algorithms are iterative in nature and seek to approximate the solution to a difficult optimization problem. The general form of an optimization problem is to

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in C, \end{aligned}$$

where $x \in \mathbb{R}^d$ is the set of d real *decision variables*, C is the *feasible set*, and $f : \mathbb{R}^d \to \mathbb{R}$ is the *objective function*. The goal is to find $x \in C$ such that $f(x)$ is as small as possible. For example, in structural mechanics, x could be lengths and widths of the members in a truss design, C could encode stress limitations of the materials and load bearing constraints, and f could be the total cost of the design. Solving this optimization problem would provide the cheapest truss design that satisfies all design specifications. In statistics, x could be parameters in a statistical model, C could correspond to constraints such as certain parameters being positive, and f could be the negative log-likelihood of the model given the observed data. Solving this optimization problem finds the maximum likelihood model. While some optimization problems can be solved analytically (such as least squares problems), analytical solutions do not exist for most optimization models, and numerical methods must be used to approximate the solution. Even when an analytical solution exists, it may be preferable to use numerical methods because they can be more computationally efficient. Iterative optimization algorithms typically begin with an estimate x^0 of the solution, and each step refines the estimate, producing a sequence $x^0, x^1, \ldots$. If properly designed, then $x^k \to x^\star$ in the limit, and as many iterations as needed can be used to achieve the desired level of accuracy. Depending on the nature and structure of f and C in the optimization problem, different optimization algorithms may be appropriate. The design, selection, and tuning of optimization algorithms is more of an art than a science. Many different algorithms have been proposed, some with theoretical guarantees and others with a strong record of empirical performance. In practice, some algorithms converge slowly yet are predictable and reliable. Meanwhile, other algorithms converge more rapidly on average, but can fail spectacularly in some cases. Algorithm selection and tuning is typically performed by experts with deep area knowledge. In many ways, iterative algorithms behave like control systems, and the choice of algorithm is akin to the choice of controller. In the sections that follow, we formalize this connection and describe how optimization algorithms can be viewed as controllers performing robust control. This work will also show how dissipativity theory can be used to analyze the performance of many classes of optimization algorithms. This allows selection and tuning of optimization algorithms to be performed in an automated and systematic way.

1 Black-box paradigm and performance evaluation

To reason about different optimization algorithms and their performance, it is common to employ a *black-box* paradigm [23, §1.1.2]. This model assumes the availability of *oracles* that can be queried to provide pertinent information about the objective function or the constraints. The oracles are how the algorithm interfaces with the optimization problem. For example, if f is differentiable and C is a convex set, a popular algorithm is *projected gradient descent*, which begins with a some guess value x^0 and follows the iterations:

$$x^{k+1} = \Pi_C\left(x^k - \eta \nabla f(x^k)\right), \qquad \text{for } k = 0, 1, \ldots \tag{1}$$

Here, $\eta > 0$ is a tuning parameter called the *stepsize*, ∇f is the gradient of f, and Π_C is the projection onto the set C. For sufficiently small η and under appropriate regularity conditions on f, projected gradient descent will converge to a solution $x^\star$ of the constrained optimization problem. This example includes two oracles:

1. Gradient oracle: Given x, return $\nabla f(x)$.
2. Projection oracle: Given x, return $\Pi_C(x)$.

In the black-box model, each oracle query incurs a fixed cost (usually *time*) and all other costs associated with computer storage or memory are ignored. The *performance* of an iterative method is based on the total time required for an error measure to reach a specified value. Common error measures include distance to optimality $\|x^k - x^\star\|$ and function error $f(x^k) - f(x^\star)$. Iterative algorithms typically call each oracle once per iteration, so performance can be evaluated by measuring the *convergence rate*, which is how quickly the error measure decreases per iteration. For example, an algorithm exhibits *geometric* convergence if there is some $\rho \in [0, 1)$ and $K > 0$ such that

$$\|x^k - x^\star\| \leq K\rho^k \|x^0 - x^\star\| \quad \text{for } k = 0, 1, \ldots. \tag{2}$$

Smaller ρ corresponds to a faster algorithm. The smallest value of ρ that satisfies (2) can depend on the oracles used (the choice of f and C) and the initial condition x^0. Care must be taken when interpreting convergence rates, because as ρ becomes smaller, K may get larger, and as ρ approaches its minimum value, then we may have $K \to \infty$.

Algorithm analysis Algorithm analysis is the practice of determining bounds on the rate of convergence of algorithms subject to various assumptions. Conventionally, algorithm analysis provides an assurance that a given algorithm will work well for many instances of an optimization problem. For example, it may be desirable for algorithm $\mathcal{A}$ to converge rapidly for many different pairs of oracles (f, C). We call this set of admissible oracle pairs $\mathcal{F}$. A typical analysis query might be: What is the *worst-case* geometric convergence rate ρ achieved by $\mathcal{A}$ over the set $\mathcal{F}$? Mathematically, this is given by the expression

$$\rho(\mathcal{A}, \mathcal{F}) := \inf\left\{ \rho > 0 \;\middle|\; \sup_{(f,C)\in\mathcal{F}} \sup_{x^0} \sup_{k\geq 0} \frac{\|x^k - x^\star\|}{\rho^k \|x^0 - x^\star\|} < \infty \right\}. \tag{3}$$

Equation (3) states that for any $\rho > \rho(\mathcal{A}, \mathcal{F})$, the geometric convergence criterion (2) holds for all choices of oracles $(f, C) \in \mathcal{F}$ and all initial conditions x^0. So, $\rho(\mathcal{A}, \mathcal{F})$ is the fastest convergence

rate that is guaranteed to hold over all admissible problem instances and initial conditions. Two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ can then be compared based on their convergence rates. If $\rho(\mathcal{A}_1, \mathcal{F}) < \rho(\mathcal{A}_2, \mathcal{F})$, then $\mathcal{A}_1$ is faster than $\mathcal{A}_2$ in the worst case. The notion of *worst-case convergence rate* in algorithm analysis is akin to the notion of *robust stability* in nonlinear control. The following sections explore this connection in greater detail and show how tools from robust control can be brought to bear on the problem of algorithm analysis.

2 Algorithm analysis as robust control

The performance evaluation of iterative algorithms under the black-box paradigm may be reframed as certifying robust stability for a feedback system. Specifically, the algorithm can be written as a discrete-time dynamical system in feedback with its oracles. It will be illustrated through examples how a variety of algorithms can be converted to *feedback form*.

2.1 Projected gradient descent

Returning to the projected gradient descent example (1), the algorithm has access to two oracles: a projection operator Π_C and the gradient ∇f. If the problem is unconstrained, the simplification $\Pi_C = I$ occurs and (1) becomes ordinary gradient descent. It is converted to feedback form by defining the input-output pairs of Π_C and ∇f as (y_1, u_1) and (y_2, u_2), respectively. The ensuing block diagram is illustrated in Figure 1.

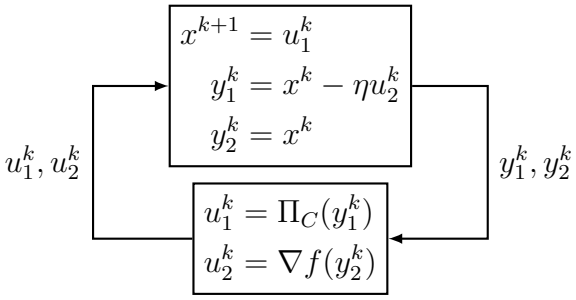


Figure 1. Feedback interconnection for projected gradient descent, an iterative algorithm with update equations given by (1). The feedback form separates the algorithm dynamics (which are a linear time-invariant (LTI) system) from the oracle calls (which are treated as unknown nonlinearities).

2.2 Nesterov's accelerated method

Nesterov's accelerated method [23, §2.2] is a popular iterative approach used to solve the unconstrained optimization problem $\min_x f(x)$, where f is continuously differentiable, and access to a gradient oracle ∇f is provided. The algorithm has two states (x^k, y^k) and uses the update

$$y^k = x^k + \beta(x^k - x^{k-1}) \tag{4a}$$

$$x^{k+1} = y^k - \eta \nabla f(y^k). \tag{4b}$$

The state y^k extrapolates based on the current iterate x^k and previous iterate x^{k-1}. Then, gradient descent is performed based on y^k. The *momentum parameter* β controls the amount of extrapolation. When $\beta = 0$, $y^k = x^k$ and the familiar gradient descent algorithm is recovered. The idea

is to tune β to obtain faster convergence than ordinary gradient descent. To convert (4) to feedback form, substitute (4a) into (4b) and define the new state variables $x_1^k = x^k$ and $x_2^k = x^{k-1}$. We then obtain the feedback form illustrated in Figure 2.

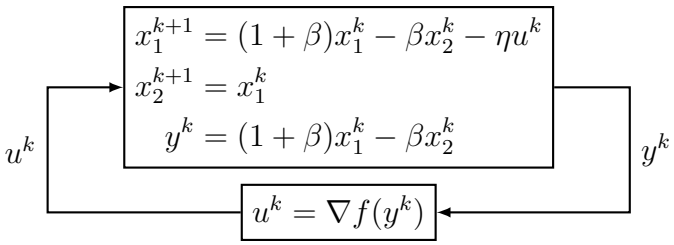


Figure 2. Feedback interconnection for Nesterov's accelerated method, whose update equations are given in (4).

2.3 ADMM algorithm

The Alternating Direction Method of Multipliers (ADMM) [5] is a popular algorithm used to solve *composite* optimization problems. That is, the objective function can be split into two parts with different properties. The canonical problem takes the form:

$$\begin{aligned} \text{minimize} \quad & f(x) + g(z) \\ \text{subject to:} \quad & Ax + Bz = c. \end{aligned}$$

For example, f may be convex and differentiable, while g may be a nondifferentiable regularization term or the indicator set for a convex constraint. The ADMM algorithm has three state variables (x^k, z^k, w^k) and a tuning parameter η. The algorithm uses the update

$$\begin{aligned} x^{k+1} &= \arg\min_x \; f(x) + \tfrac{1}{2\eta}\|Ax + Bz^k - c + w^k\|^2, \\ z^{k+1} &= \arg\min_z \; g(z) + \tfrac{1}{2\eta}\|Ax^{k+1} + Bz - c + w^k\|^2, \\ w^{k+1} &= w^k + Ax^{k+1} + Bz^{k+1} - c. \end{aligned}$$

To simplify exposition, consider the composite unconstrained case: $\min_x f(x) + g(x)$. This is the special case with $A = I$, $B = -I$, and $c = 0$. The ADMM algorithm can be written in feedback form in several equivalent ways, depending on which oracles are used. For example, the *proximal operator* [25] is defined as $\text{prox}_f(z) := \arg\min_x f(x) + \frac{1}{2}\|x - z\|^2$. Use it to rewrite the ADMM update equations as

$$x^{k+1} = \text{prox}_{\eta f}(z^k - w^k), \tag{5a}$$

$$z^{k+1} = \text{prox}_{\eta g}(x^{k+1} + w^k), \tag{5b}$$

$$w^{k+1} = w^k + x^{k+1} - z^{k+1}. \tag{5c}$$

Alternatively, if f is differentiable and g is convex but not differentiable, replace the prox updates by their corresponding first-order optimality conditions. This yields

$$0 = \nabla f(x^{k+1}) + \tfrac{1}{\eta}(x^{k+1} - z^k + w^k), \tag{6a}$$

$$0 \in \partial g(z^{k+1}) + \tfrac{1}{\eta}(z^{k+1} - x^{k+1} - w^k), \tag{6b}$$

$$w^{k+1} = w^k + x^{k+1} - z^{k+1}, \tag{6c}$$

where $\partial g(z) := \left\{v \in \mathbb{R}^d \;\middle|\; g(x) - g(z) \geq v^\mathsf{T}(x - z) \text{ for all } z \in \mathbb{R}^d\right\}$ denotes the set of subgradients of g. If g is differentiable, then $\partial g(z) = \{\nabla g(z)\}$. Both (5) and (6) can be put in feedback form to obtain the block diagrams illustrated in Figure 3. In both cases, the state variable x^k can be eliminated, so only two states are needed. The ADMM representation that uses gradients and subgradients in Figure 3 is *implicit* because it contains a circular dependency: y_1^k depends on u_1^k, which in turn depends on y_1^k. Therefore, this feedback representation cannot be used as a substitute for an implementation such as (5). Nevertheless, the implicit representation can still be used in dissipativity theory for algorithm analysis.

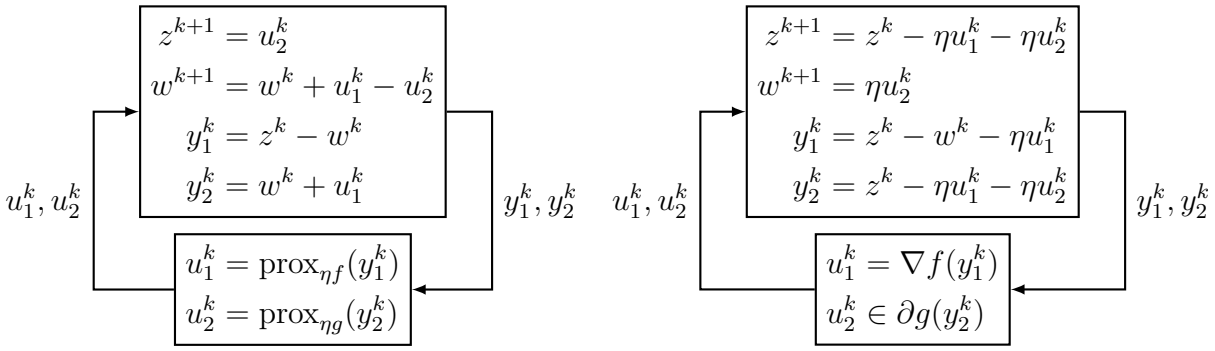


Figure 3. Equivalent feedback interconnections for the Alternating Direction Method of Multipliers (ADMM) applied to composite unconstrained optimization. **Left:** An explicit loop that uses proximal operators as oracles (5). **Right:** An implicit loop that uses gradient and subgradient oracles (6). Other representations are possible, for example using ∇f and $\mathrm{prox}_{\eta g}$. Any of these representations of ADMM can be used for analysis in the dissipativity framework and yield the same results.

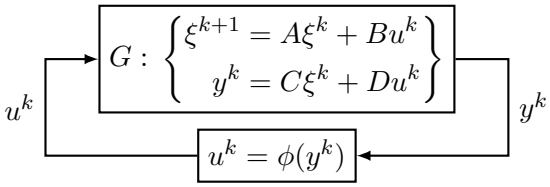
2.4 More general algorithms

In all the cases above, the algorithms can be expressed in the form of a linear time-invariant (LTI) system G in feedback with the oracles. Letting ξ^k, u^k, and y^k be the concatenated states, oracle outputs, and oracle inputs, the algorithms are represented in the following general form.

$$\xi^{k+1} = A\xi^k + Bu^k \tag{7a}$$

$$y^k = C\xi^k + Du^k \tag{7b}$$

$$u^k = \begin{bmatrix} u_1^k \\ \vdots \\ u_m^k \end{bmatrix} = \begin{bmatrix} \phi_1(y_1^k) \\ \vdots \\ \phi_m(y_m^k) \end{bmatrix} = \phi(y^k). \tag{7c}$$



The fact that G is LTI will be important for our dissipativity analysis, as it will allow us to search for Lyapunov functions in a tractable manner. Not all algorithms have a feedback form with an LTI G. For example,

- Algorithms with parameters that change on a fixed schedule, such as gradient descent with a diminishing stepsize, will yield a feedback representation where G is a linear time-varying (LTV) system.
- Algorithms with parameters that change adaptively, such as the nonlinear conjugate gradient method, will yield a feedback representation where G is a linear parameter-varying (LPV) system.
- Algorithms where the state updates are not linear functions of the previous state or oracle outputs will yield a feedback representation where G is nonlinear.

Despite these limitations, algorithms can still generally be written as a feedback interconnection of some system G and the set of oracles (nonlinearities). Since dissipativity theory can be applied to any system [36] (including LTV, LPV, and systems with nonlinear dynamics), the dissipativity approach can in principle be used to analyze any iterative algorithm.

3 Dissipativity theory

Dissipativity theory may be viewed as a counterpart to Lyapunov theory but for systems with inputs. Consider a discrete-time dynamical system satisfying the state-space equation

$$\xi^{k+1} = A\xi^k + Bu^k.$$

In classical dissipativity theory [36,37], u^k is an external supply that drives the dynamics governed by the state ξ^k. For example, in a mechanical system, u^k would be a vector of external forces and torques, while ξ^k would be a vector of generalized coordinates such as positions and velocities. The two key concepts in dissipativity are *storage* and *supply*.

1. The *storage function* $V(\xi^k)$ can be interpreted as a notion of stored energy. In our mechanical example, this would be the total energy (kinetic and potential) in the system. The storage function always satisfies $V(\xi^k) \geq 0$.
2. The *supply rate* $S(\xi^k, u^k)$ can be interpreted as a notion of work done by the external forces and torques. The supply rate may also depend on the current values of the generalized coordinates. When $S > 0$, the external force is *adding* energy to the system. When $S < 0$, the external force is *extracting* energy from the system.

A *dissipation inequality* states that the change in stored energy can be no greater than the energy provided by the external supply:

$$V(\xi^{k+1}) - V(\xi^k) \leq S(\xi^k, u^k). \tag{8}$$

If a dissipation-like inequality holds and the supply rate is of a particular form, useful stability properties of the system can be deduced. We now consider different types of supply rates that are relevant for optimization algorithms.

4 Supply rates for families of oracles

In the context of robust control, the most well-studied classes of nonlinearities include sector-bounded and slope-restricted nonlinearities, because they can be used to model common nonlinear phenomena such as saturation and stiction. In the context of optimization algorithms, oracles can often have similar properties, thus creating parallels between the two application areas. For each type of nonlinearity or oracle, we describe how to formulate an appropriate supply rate that can be used to analyze optimization algorithms.

4.1 Sector-bounded oracle

An oracle $u = \phi(y)$ is *sector-bounded* with lower bound m and upper bound L with $m < L$ if the input-output pair (y, u) satisfies

$$S_{\mathcal{C}}(y,u) := \begin{bmatrix} y \\ u \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} mL & -\frac{L+m}{2} \\ -\frac{L+m}{2} & 1 \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix} \leq 0. \tag{9}$$

In the general dissipativity framework, the supply rate $S(\xi, u)$ may depend on the state ξ and the input u. The form of the supply rate $S_{\mathcal{C}}$ in (9) still fits into this framework because the oracle is a function of y, which will depend on ξ and u through the algorithm update equations (7). In this case, $S_{\mathcal{C}} \leq 0$. Thus, in the energy interpretation of dissipativity, the *external force* is extracting energy from the system. The inequalities (9) hold *pointwise* in time for any pair (y, u), so the graph of a sector-bounded oracle is contained in the *interior conic* region illustrated in Figure 4. Denote the set of all sector-bounded nonlinearities with parameters (m, L) with the symbol $\mathcal{C}_{m,L}$. A useful pair of inequalities that follow from (9) are given by

$$m\|y\| \leq \|u\| \leq L\|y\| \tag{10}$$

$$m\|y\|^2 \leq u^{\mathsf{T}} y \leq L\|y\|^2. \tag{11}$$

See B for insight on how inequalities such as (10)–(11) can be proved. Sector-bounded nonlinearities are widely studied in controls, dating back to the introduction of *absolute stability* by Lur'e and Postnikov [19]. In this setting, ϕ is a static nonlinearity. Special cases include the small-gain theorem and passivity theory [38], where $\|\phi(y)\| \leq \gamma\|y\|$ ($m = -\gamma$ and $L = \gamma$) and $y^{\mathsf{T}}\phi(y) \geq 0$ ($m = 0$ and $L \to \infty$), respectively. Sector-bounded oracles can occur in optimization when oracles are subject to multiplicative noise. For example, round-off error may occur due to finite-precision arithmetic. Alternatively, a call to an oracle may involve a complicated simulation that inherently produces approximate results due to time budget limitations. If y is the true signal, multiplicative noise transforms the signal into $u = y + \delta_y$, where $\|\delta_y\| \leq \varepsilon\|y\|$. Here, ε is the noise strength ($\varepsilon = 0.1$ would correspond to 10% multiplicative noise). Rearranging this inequality yields $\|y - u\| \leq \varepsilon\|y\|$. Comparing to (9), this corresponds to a sector-bounded nonlinearity with $m = 1 - \varepsilon$ and $L = 1 + \varepsilon$.

4.2 Slope-restricted oracles

An oracle $u = \phi(y)$ is *slope-restricted* with lower bound m and upper bound L with $m < L$ if every pair of input-output pairs (y_1, u_1) and (y_2, u_2) satisfies

$$S_{\mathcal{M}}(y_1, y_2, u_1, u_2) := \begin{bmatrix} y_2 - y_1 \\ u_2 - u_1 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} mL & -\frac{L+m}{2} \\ -\frac{L+m}{2} & 1 \end{bmatrix} \begin{bmatrix} y_2 - y_1 \\ u_2 - u_1 \end{bmatrix} \leq 0. \tag{12}$$

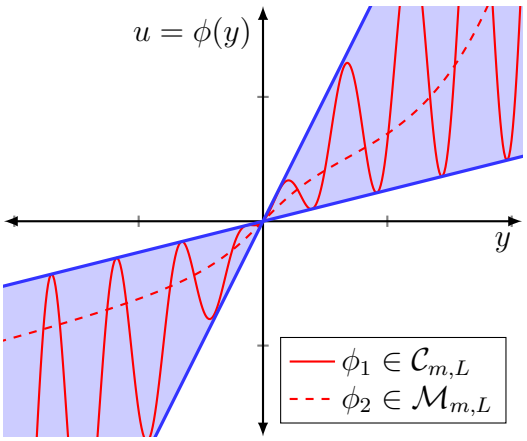


Figure 4. One-dimensional examples of a sector-bounded nonlinearity $\phi_1 \in \mathcal{C}_{m,L}$ and a slope-restricted nonlinearity $\phi_2 \in \mathcal{M}_{m,L}$, both with $m = \frac{1}{4}$ and $L = 2$. In the sector-bounded case, all input-output pairs (u, y) satisfy the quadratic inequality (9) (which is $\frac{1}{2}y^2 - \frac{9}{4}yu + u^2 \leq 0$), shown as the shaded region. In the slope-restricted case, a stronger condition is satisfied: $\frac{1}{2}(y_1 - y_2)^2 - \frac{9}{4}(y_1 - y_2)(u_1 - u_2) + (u_1 - u_2)^2 \leq 0$, for all $y_1, y_2, u_1, u_2 \in \mathbb{R}$. In other words, the slope of the line connecting any pair of points on the graph must be between $\frac{1}{4}$ and 2.

In a manner analogous to how (10) and (11) were derived for sector-bounded nonlinearities, it can be shown that slope-restricted nonlinearities enjoy the properties

$$m\|y_2 - y_1\| \leq \|u_2 - u_1\| \leq L\|y_2 - y_1\| \tag{13}$$

$$m\|y_2 - y_1\|^2 \leq (u_2 - u_1)^\mathsf{T}(y_2 - y_1) \leq L\|y_2 - y_1\|^2. \tag{14}$$

Slope-restricted nonlinearities (also called *incremental*) were studied by Zames and Falb [39] (for example, *incremental passivity* or *incremental small-gain*). Examples of slope-restricted nonlinearities include saturation or elements exhibiting hysteresis. See Figure 4 for a visual example. Denote the set of all slope-restricted nonlinearities with parameters (m, L) with the symbol $\mathcal{M}_{m,L}$. In optimization, different types of slope-restrictedness bear different names. The proximal operator prox_g for any convex function g, used for example in ADMM (5), is *firmly nonexpansive*. That is, $(x - y)^\mathsf{T}(\mathrm{prox}_g(x) - \mathrm{prox}_g(y)) \geq \|\mathrm{prox}_g(x) - \mathrm{prox}_g(y)\|^2$ for all $x, y \in \mathbb{R}^d$ [1, Prop. 4.16 and 12.28]. In other words, prox_f satisfies (12) with $m = 0$ and $L = 1$. A special case is when g is the indicator function of a convex set C [that is $g(x) = 0$ if $x \in C$ and $g(x) = +\infty$ otherwise], then $\mathrm{prox}_g(x) = \Pi_C(x)$ is the Euclidean projection of x onto the set C. Another example is the subgradient of a convex function g, which were used in (6). Subgradients are *monotone*, that is, $(x_1 - x_2)^\mathsf{T}(v_1 - v_2) \geq 0$ for all $x_i \in \mathbb{R}^d$ and $v_i \in \partial g(x_i)$. In other words, ∂g satisfies (12) with $m = 0$ and $L \to \infty$. A special case is when g is quadratic and positive semidefinite: $g(x) = x^\mathsf{T}Ax$, where $A + A^\mathsf{T} \succeq 0$.

4.3 Gradient of a convex function

The case where the nonlinearity is the gradient of a convex and continuously differentiable function is of particular interest in the field of optimization, even if it is a less common occurrence in controls. If f is a continuously differentiable function, define the following notions.

1. f is *strongly convex* with parameter $m > 0$ if $g(x) := f(x) - \frac{m}{2}\|x\|^2$ is a convex function. In other words, $\theta g(x) + (1 - \theta)g(y) \geq g(\theta x + (1 - \theta)y)$ for all $x, y \in \mathbb{R}^d$ and $\theta \in [0, 1]$.

2. f has *Lipschitz gradients* with parameter $L > 0$ if $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^d$.

Strong convexity means that f is not too flat, while Lipschitz gradients ensure that f does not grow too quickly. We now state a useful property of functions that are both strongly convex and have Lipschitz gradients. Denote the set of all nonlinearities satisfying the two properties above with the symbol $\mathcal{F}_{m,L}$.

Theorem 1. *Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable, strongly convex with parameter m, and has Lipschitz gradients with parameter L. For all $y_i \in \mathbb{R}^d$ with $f_i = f(y_i)$ and $u_i = \nabla f(y_i)$ for $i = 1, 2$,*

$$\mathcal{S}_{\mathcal{F}}(y_1, y_2, u_1, u_2) := \frac{1}{2(L-m)} \begin{bmatrix} y_2 - y_1 \\ u_2 - u_1 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} mL & -\frac{L+m}{2} \\ -\frac{L+m}{2} & 1 \end{bmatrix} \begin{bmatrix} y_2 - y_1 \\ u_2 - u_1 \end{bmatrix} + \frac{1}{2}(u_1 + u_2)^{\mathsf{T}}(y_2 - y_1) \leq f_2 - f_1. \quad (15)$$

This result is proven for example in [32, Thm. 4]. In large-scale optimization problems, it is typical to assume the availability of a gradient oracle. Algorithms that use such an oracle are called *first-order methods*. Popular examples of optimization problems that are often solved on a large scale include logistic regression and regularized least-squares problems such as the lasso [15].

4.4 Other oracle types

Many optimization problems involve objective functions (oracles) that are convex, but not strongly convex. This has motivated the study of conditions that are weaker than strong convexity but can still provide useful convergence guarantees. One such example is the *Polyak–Łojasiewicz* condition: $\frac{1}{2m}\|u\|^2 \geq f - f^\star$. Others include the *error bound* condition, the *quadratic growth* condition, and the *restricted secant inequality*. For a survey of such conditions, refer to [3, 16] and references therein. Although we restrict attention to strong convexity in the present article, the aforementioned alternatives to strong convexity can be used just as easily. All of these conditions are characterized by inequalities similar to (15) that can be used directly as supply rates; they are quadratic in $(x^k - x^\star)$ and u^k, and they are linear in $f^k - f^\star$.

4.5 Nestedness and supply rates

The three classes of nonlinearities characterized by (9), (12), and (15) are nested. That is, $\mathcal{F}_{m,L} \subseteq \mathcal{M}_{m,L} \subseteq \mathcal{C}_{m,L}$.This follows because if (15) is added to itself with indices interchanged, then (12) is recovered. So, (12) is a special case of (15). Moreover, if $y_1 = 0$ (and $u_1 = 0$) in (12), then (9) is recovered. So, (9) is a special case of (12). In the one-dimensional case ($d = 1$), all gradients of convex functions are slope-restricted, and vice versa. In other words, $\mathcal{F}_{m,L} = \mathcal{M}_{m,L}$. This is not the case when $d \geq 2$, which is related to the notion of cyclic monotonicity [27]. The nestedness property implies that a supply rate that is valid for one class of oracles is also valid for any oracle belonging to a subclass. For each class of oracles, the fundamental inequalities (9), (12), and (15) can be directly used as supply rates. Typically, there will be many valid choices of supply rates, and different choices could yield different convergence rate guarantees. The case studies that follow show how to optimize the choice of supply rate to produce the least conservative estimate of convergence rate attainable.

5 Certifying convergence rates using dissipativity

5.1 Certifying geometric convergence

Geometric convergence, also known as *exponential stability* in the controls literature and *linear convergence* in the optimization literature, can be verified using a modified dissipation inequality of the form

$$V(\xi^{k+1}) - qV(\xi^k) \leq S(\xi^k, u^k), \tag{16}$$

where $0 \leq q < 1$. We distinguish between two important cases for use in algorithm analysis.

1. If the supply rate is nonpositive, $S(\xi^k, u^k) \leq 0$, then the dissipation inequality implies that $V(\xi^{k+1}) \leq qV(\xi^k)$. So, V decreases by a factor of q at every timestep, which means that V is a Lyapunov function that certifies geometric convergence. This case will be used when $\nabla f \in \mathcal{C}_{m,L}$ or $\nabla f \in \mathcal{M}_{m,L}$.

2. If the supply rate satisfies the more general condition $S(\xi^k, u^k) \leq q\psi(\xi^k) - \psi(\xi^{k+1})$, where ψ is a nonnegative function, then (16) can be rewritten as $V(\xi^{k+1}) + \psi(\xi^{k+1}) \leq q\left(V(\xi^k) + \psi(\xi^k)\right)$. In other words, $V(x) + \psi(x)$ is a Lyapunov function that certifies geometric convergence. This case will be used when $\nabla f \in \mathcal{F}_{m,L}$.

5.2 Certifying other convergence rates

Dissipativity can also be used to certify other rates of convergence. For example, consider gradient descent (the algorithm state is $\xi^k := x^k$), where the function value $f(x^k)$ decreases at each iteration. If the standard dissipation inequality (8) holds with a supply rate that satisfies $S(x^k, u^k) \leq -\big(f(x^k) - f(x^\star)\big)$, then the dissipation inequality implies that $V(x^{k+1}) - V(x^k) \leq -\big(f(x^k) - f(x^\star)\big)$. Summing over k,

$$f(x^k) - f(x^\star) \leq \frac{1}{k+1} \sum_{i=0}^{k} \big(f(x^i) - f(x^\star)\big) \leq \frac{1}{k+1} \Big(V(x^0) - V(x^{k+1})\Big) \leq \frac{1}{k+1} V(x^0).$$

In other words, the algorithm converges in *function value*, and the function value decreases at the *sublinear rate* $1/k$, which is slower than geometric convergence. Dissipativity can also be used to certify other sublinear rates, such as $1/k^2$ [11]. Ultimately, dissipation inequalities lead to Lyapunov functions. However, dissipation inequalities also provide a framework by which to treat uncertain disturbance inputs. Roughly, this is done by using supply rates that are satisfied by the disturbances and then searching for storage functions such that a dissipation inequality is satisfied (thereby certifying robust stability). Even if the system in question is not mechanical in nature or there is no clear notion of *energy*, the system may still satisfy a dissipation inequality. This is akin to the idea of that Lyapunov functions can be used to certify the stability of an autonomous differential or difference equation, even when the equation does not describe a physical system. In the context of algorithm analysis, we view the outputs of the oracles as disturbance inputs for the iterative algorithm in question and use dissipativity theory to certify robust convergence properties.

6 Case study: Gradient Descent

Gradient descent is perhaps the most recognizable iterative algorithm. We will now illustrate different approaches for analyzing gradient descent for a simple class of functions. Consider the

(unconstrained) problem of minimizing $f(x)$, where f is m-strongly convex and has L-Lipschitz gradients. That is, $f \in \mathcal{F}_{m,L}$. Gradient descent is defined by the iteration

$$x^{k+1} = x^k - \eta \nabla f(x^k) \qquad \text{for } k = 0, 1, \ldots. \tag{17}$$

Our task is to find the worst-case linear convergence rate $\rho(\mathcal{A}, \mathcal{F})$ and ultimately the choice of stepsize η that leads to the fastest convergence rate.

Nesterov's analysis A classical approach to analyzing gradient descent is due to Nesterov [23, Thm. 2.1.15] and begins by bounding the error at the $(k+1)^{\text{st}}$ iterate in terms of the error at the k^{th} iterate:

$$\begin{aligned}
\|x^{k+1} - x^\star\|^2 &\overset{(17)}{=} \|x^k - x^\star - \eta \nabla f(x^k)\|^2 \\
&= \|x^k - x^\star\|^2 + \eta^2 \|\nabla f(x^k)\|^2 - 2\eta \left(x^k - x^\star\right)^{\mathsf{T}} \nabla f(x^k) \\
&\overset{(9)}{\leq} \|x^k - x^\star\|^2 + \eta^2 \|\nabla f(x^k)\|^2 - \tfrac{2\eta}{L+m} \left(mL\|x^k - x^\star\|^2 + \|\nabla f(x^k)\|^2\right) \\
&= \left(1 - \tfrac{2\eta m L}{L+m}\right) \|x^k - x^\star\|^2 + \eta \left(\eta - \tfrac{2}{L+m}\right) \|\nabla f(x^k)\|^2.
\end{aligned} \tag{18}$$

If it is further assumed that $0 \leq \eta \leq \frac{2}{L+m}$, the second term in (18) is nonpositive, and it can be concluded that the error shrinks at every iteration according to

$$\|x^{k+1} - x^\star\| \leq \sqrt{1 - \tfrac{2\eta m L}{L+m}}\, \|x^k - x^\star\|.$$

The contraction factor $\sqrt{1 - \frac{2\eta m L}{L+m}}$ is an upper bound on the worst-case convergence rate. If the stepsize $\eta \in [0, \frac{2}{L+m}]$ is selected to minimize this upper bound, the optimal stepsize is $\eta = \frac{2}{L+m}$ and yields the bound $\rho \geq \frac{L-m}{L+m}$ on the worst-case convergence rate.

Polyak's analysis Another approach to analyzing gradient descent is due to Polyak [26, §1.4, Thm. 3]. Assume that f is twice differentiable. By the fundamental theorem of calculus,

$$\nabla f(x^k) = \nabla f(x^\star) + \int_0^1 \nabla^2 f(x^\star + \tau(x^k - x^\star))(x^k - x^\star)\, \mathrm{d}\tau = A_k(x^k - x^\star),$$

where $A_k := \int_0^1 \nabla^2 f(x^\star + \tau(x^k - x^\star))\, \mathrm{d}\tau$. Then, we can bound the error at the $(k+1)^{\text{st}}$ iterate in terms of the error at the k^{th} iterate using the triangle inequality:

$$\|x^{k+1} - x^\star\| = \|x^k - x^\star - \eta \nabla f(x^k)\| = \|(I - \eta A_k)(x^k - x^\star)\| \leq \|I - \eta A_k\| \cdot \|x^k - x^\star\|.$$

It follows from (14) that $mI \preceq \nabla^2 f(x) \preceq LI$ for all x. Therefore, $mI \preceq A_k \preceq LI$. Since $I - \eta A_k$ is symmetric, its norm can be bounded in terms of the smallest and largest eigenvalues of A_k, which gives a bound on the worst-case convergence rate:

$$\rho \geq \|I - \eta A_k\| = \max\left\{|1 - \eta m|, |1 - \eta L|\right\}.$$

This bound can be minimized if $\eta = \frac{2}{L+m}$, which yields $\rho \geq \frac{L-m}{L+m}$. Both the Nesterov and Polyak analyses of gradient descent yield the same optimal stepsize of $\eta = \frac{2}{L+m}$ and the same worst-case

convergence rate bound $\rho \geq \frac{L-m}{L+m}$. However, the bounds on ρ disagree when $\eta \neq \frac{2}{L+m}$; Nesterov's bound is more conservative than Polyak's. Conversely, Nesterov's approach only assumed ∇f was sector-bounded, while Polyak made the stronger assumptions that f is twice differentiable and ∇f is slope-restricted. It is possible to prove Polyak's bound without assuming twice-differentiability, but it requires a different approach.

Dissipativity approach The gradient method can also be analyzed using dissipativity theory. First write the gradient method as a linear time-invariant dynamical system in feedback with ∇f:

$$x^{k+1} = x^k - \eta u^k, \tag{19a}$$

$$y^k = x^k, \tag{19b}$$

$$u^k = \nabla f(y^k). \tag{19c}$$

We then use the supply rate $S_{\mathcal{C}}$ associated with sector-bounded nonlinearities, given by (9). The sector bound will not hold as written in (9) because the oracle $u = \nabla f(y)$ is zero when $y = x^\star = y^\star$ (the solution of the optimization problem), not when $y = 0$. To account for this, use the inequality $S_{\mathcal{C}}(y - y^\star, u) \leq 0$. Although $y^\star$ may not be known in advance, the dissipativity approach only requires assuming existence of $y^\star$ and does not depend on its actual value. The idea is to use a quadratic Lyapunov function candidate $V(x) = \|x - x^\star\|^2$ and to certify a dissipation inequality of the form

$$V(x^{k+1}) - \rho^2 V(x^k) \leq \lambda S_{\mathcal{C}}(y^k - y^\star, u^k), \tag{20}$$

Where $\lambda \geq 0$. If (20) holds, then $S_{\mathcal{C}}(y^k - y^\star, u^k) \leq 0$ implies that $V(x^{k+1}) \leq \rho^2 V(x^k)$, and therefore $\|x^{k+1} - x^\star\| \leq \rho \|x^k - x^\star\|$ and ρ is an upper bound on the worst-case convergence rate. Substituting the definitions for V and $S_{\mathcal{C}}$ and the dynamics (19) into (20),

$$\|x^k - x^\star - u^k\|^2 - \rho^2 \|x^k - x^\star\| \leq \lambda (m(x^k - x^\star) - u^k)^{\mathsf{T}} (L(x^k - x^\star) - u^k).$$

This quadratic expression in $x^k - x^\star$ and u^k must hold for all choices of $x^k - x^\star$ and u^k, which implies the inequalities

$$\begin{bmatrix} 1 - \rho^2 - \lambda m L & -\eta + \lambda \frac{L+m}{2} \\ -\eta + \lambda \frac{L+m}{2} & \eta^2 - \lambda \end{bmatrix} \preceq 0, \qquad \lambda \geq 0, \tag{21}$$

where "$\preceq$" denotes inequality in the semidefinite sense. The inequalities (21) do not depend explicitly on $y^\star$, even though the existence of $y^\star$ is assumed as part of the derivation. The task of minimizing ρ subject to (21) is a semidefinite program (SDP). While typically solved numerically, SDPs can often be solved analytically in cases such as this one, where the matrices involved are small. In this case, a Schur complement of (21) is used to obtain the equivalent inequalities:

$$\rho^2 \geq 1 - \lambda m L + \frac{(\eta - \lambda \frac{L+m}{2})^2}{\lambda - \eta^2}, \qquad \lambda \geq \eta^2. \tag{22}$$

Extremizing (22) with respect to λ yields $\rho \geq \max\{|1 - \eta m|, |1 - \eta L|\}$, which is the same bound as in Polyak's analysis. For the simple case of gradient descent, the dissipativity approach produces the tightest possible bound (same as Polyak's bound), yet it only assumes ∇f is sector-bounded. A nice feature of the dissipativity approach is that it is *systematic*. When analyzing increasingly complicated algorithms, it becomes increasingly difficult to obtain useful convergence rate bounds via the traditional approach of ad hoc equation manipulation. The dissipativity approach provides a principled and scalable way to analyze algorithms.

7 Case study: Nesterov's Accelerated Method

Consider Nesterov's accelerated method applied to a continuously differentiable function f, with access to the oracle ∇f. As in the gradient descent example, we will consider f that is m-strongly convex with L-Lipschitz gradients. That is, $f \in \mathcal{F}_{m,L}$. It will be shown that the dissipativity approach can find the tightest known convergence rate for this algorithm. Recall that Nesterov's accelerated method is characterized by the iteration

$$x_1^{k+1} = (1+\beta)x_1^k - \beta x_2^k - \eta u^k, \tag{23a}$$
$$x_2^{k+1} = x_1^k, \tag{23b}$$
$$y^k = (1+\beta)x_1^k - \beta x_2^k. \tag{23c}$$

The classical approach to analyzing Nesterov's accelerated method is *estimate sequences* [23, §2.2.1], which are a recursively generated sequence of quadratic bounds on the worst-case convergence rate ρ. The approach is rather involved, so the exposition is omitted. The net result is that the worst-case convergence rate satisfies the bound

$$\rho \geq \sqrt{1 - \sqrt{\tfrac{m}{L}}}. \tag{24}$$

We showcase the versatility of the dissipativity approach by analyzing the cases where ∇f belongs to $\mathcal{C}_{m,L}$, $\mathcal{M}_{m,L}$, or $\mathcal{F}_{m,L}$. In all of these cases, as in the analysis of gradient descent, we use a change of variables to shift the optimal point to zero. That is, the dynamics are re-expressed in terms of $x^k - x^\star$ and $y^k - y^\star$. Since the convergence rate bound found via dissipativity is independent of the optimal point, the notation is simplified by assuming without loss of generality that the optimal point is at zero, so $x^\star = y^\star = u^\star = 0$.

Nesterov for sector-bounded gradients For the sector-bounded case, the same approach is used as in the analysis of gradient descent. There are two states, x_1 and x_2, so the storage function will be a positive-definite quadratic function of both:

$$V(x) = x^\mathsf{T} P x, \quad \text{where } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ and } P \succ 0.$$

The supply rate $S_\mathcal{C}$ defined in (9) is used, and we seek the smallest ρ such that the following dissipation inequality is satisfied for some $\lambda \geq 0$:

$$V(x^{k+1}) - \rho^2 V(x^k) \leq \lambda S_\mathcal{C}(y^k, u^k). \tag{25}$$

Since the presence of both P and λ renders this dissipation inequality homogeneous, it can be normalized by setting $\lambda = 1$. Upon substituting the dynamics (23), the dissipation inequality (25) becomes a quadratic inequality in (x^k, u^k), which leads to a semidefinite program (as in the case study for gradient descent).

Nesterov for slope-restricted gradients For the slope-restricted case, the aim is to use the supply rate $S_\mathcal{M}$ defined in (12). However, using consecutive iterates as the two points in the storage function [for example (y^k, u^k) and (y^{k+1}, u^{k+1})] will cause the dissipation inequality to depend on

both u^k and u^{k+1}. Therefore, the storage function must be augmented to depend on both x^k and x^{k+1}. For the supply rate, there are many choices. In particular, apply (12) at any pair of points chosen among $\{y^k, y^{k+1}, y^\star\}$. Since the expression for $S_{\mathcal{M}}$ is symmetric in its arguments, this leads to three supply rate inequalities:

$$S^1_{\mathcal{M}} := S_{\mathcal{M}}(y^k, y^{k+1}, u^k, u^{k+1}) \leq 0, \qquad S^2_{\mathcal{M}} := S_{\mathcal{M}}(y^k, y^\star, u^k, u^\star) \leq 0,$$
$$S^3_{\mathcal{M}} := S_{\mathcal{M}}(y^{k+1}, y^\star, u^{k+1}, u^\star) \leq 0.$$

Ultimately, the dissipation inequality is

$$V(x^{k+1}, x^{k+2}) - \rho^2 V(x^k, x^{k+1}) \leq \sum_{i=1}^{3} \lambda_i S^i_{\mathcal{M}}, \tag{26}$$

where $\lambda_1, \lambda_2, \lambda_3 \geq 0$ and $V(\cdot, \cdot)$ is a positive-definite quadratic. Upon substituting the dynamics (23), the dissipation inequality (26) becomes a quadratic inequality in (x^k, u^k, u^{k+1}), which again leads to a semidefinite program. Further augmenting the storage function to include more consecutive iterates $\{x^k, x^{k+1}, \ldots, x^{k+r}\}$ will further increase the number of supply rate inequalities available for use, potentially yielding less conservative upper bounds on the worst-case convergence rate ρ.

Nesterov for gradients of strongly convex functions For the case $\nabla f \in \mathcal{F}_{m,L}$, the aim is to use the supply rate $S_{\mathcal{F}}$ defined in (15). For brevity, we write $f^k := f(y^k)$ and $f^\star := f(y^\star)$. This time, the supply rate is not symmetric in its arguments, so there are six inequalities to choose from:

$$S^1_{\mathcal{F}} := S_{\mathcal{F}}(y^{k+1}, y^k, u^{k+1}, u^k) \leq f^k - f^{k+1}, \qquad S^2_{\mathcal{F}} := S_{\mathcal{F}}(y^k, y^{k+1}, u^k, u^{k+1}) \leq f^{k+1} - f^k,$$
$$S^3_{\mathcal{F}} := S_{\mathcal{F}}(y^k, y^\star, u^k, u^\star) \leq f^\star - f^k, \qquad S^4_{\mathcal{F}} := S_{\mathcal{F}}(y^\star, y^k, u^\star, u^k) \leq f^k - f^\star,$$
$$S^5_{\mathcal{F}} := S_{\mathcal{F}}(y^{k+1}, y^\star, u^{k+1}, u^\star) \leq f^\star - f^{k+1}, \qquad S^6_{\mathcal{F}} := S_{\mathcal{F}}(y^\star, y^{k+1}, u^\star, u^{k+1}) \leq f^{k+1} - f^\star.$$

In this context, the storage function V will not itself be a Lyapunov function. Rather, the Lyapunov function will be of the form $V(\cdot) + f^k$. Therefore, the storage function V need not be positive definite. We therefore use two inequalities:

$$V(x^{k+1}, x^{k+2}) - \rho^2 V(x^k, x^{k+1}) \leq \sum_{i=1}^{6} \lambda_i S^i_{\mathcal{F}}, \tag{27a}$$

$$-V(x^k, x^{k+1}) \leq \sum_{i=1}^{6} \mu_i S^i_{\mathcal{F}}, \tag{27b}$$

where the λ_i and μ_i are nonnegative. Ultimately, the aim is for the supply rates to satisfy

$$\sum_{i=1}^{6} \lambda_i S^i_{\mathcal{F}} \leq \rho^2 \left(f^k - f^\star\right) - \left(f^{k+1} - f^\star\right), \tag{28a}$$

$$\sum_{i=1}^{6} \mu_i S^i_{\mathcal{F}} \leq \left(f^k - f^\star\right), \tag{28b}$$

which is ensured via the additional linear constraints

$$\begin{aligned} \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 &= \rho^2, & -\lambda_1 + \lambda_2 - \lambda_5 + \lambda_6 &= -1, \\ \mu_1 - \mu_2 - \mu_3 + \mu_4 &= 1, & -\mu_1 + \mu_2 - \mu_5 + \mu_6 &= 0. \end{aligned}$$

Combining (27) with (28), it follows that $V(x^k, x^{k+1}) + (f^k - f^\star)$ is a Lyapunov function that certifies geometric convergence with rate ρ. As with the case $\mathcal{M}_{m,L}$, it is possible to further augment the storage function and use more supply rates, which can potentially yield less conservative upper bounds on the worst-case convergence rate ρ.

Numerical simulation When examining the semidefinite programs (25), (26), and (27), they differ from the gradient descent case (21) in that they are not linear in ρ^2 due to the presence of the product $\rho^2 P$. However, they are linear in P and the λ_i's and μ_i's for each fixed ρ, so they can be solved by bisection on ρ. Implementing these solutions with the default tuning of Nesterov's method, which is $\eta = \frac{1}{L}$ and $\beta = \frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}$, we obtain the results displayed in Figure 5. These results mirror those reported in [18], although that article used the theory of integral quadratic constraints (IQCs) [21] and Zames–Falb multipliers [10, 40] instead of dissipativity. IQC theory is closely related to dissipativity theory [30], and the dissipativity approach presented above is algebraically equivalent to the approach used in [18]. As shown in Figure 5, the worst-case rate certified by the dissipativity approach improves upon the rate found using the classical approach of estimate sequences. Figure 5 also plots the *iteration complexity*, which is the number of iterations required to reach a certain error ε. The iteration complexity is proportional to $-1/\log\rho$. The results in Figure 5 were obtained numerically using CVX [9], a Matlab package for specifying and solving convex programs. In this case, the semidefinite program is more complicated than the one for gradient descent, yet still simple enough to be solved analytically [28]. When using the Lyapunov function $V(x^{k+1}, x^k) + (f^k - f^\star)$, it is not required that V itself be positive, nor is it required that V or f^k be monotonically decreasing. Figure 6 shows two examples of Nesterov's method applied to functions in $\mathbb{R}^2$. Both cases use functions with $\nabla f \in \mathcal{F}_{m,L}$ with the same values of (m, L) and the same initialization. The functions used are

$$f_1(x_1, x_2) := \frac{m}{2}(x_1^2 + x_2^2) + (L - m)\log\left(e^{-x_1} + e^{x_1/3 + x_2} + e^{x_1/3 - x_2}\right) \tag{29a}$$

$$f_2(x_1, x_2) := \tfrac{L}{2}x_1^2 + \tfrac{m}{2}x_2^2. \tag{29b}$$

In the first case (left panel of Figure 6), the distance to optimality $\|x^k - x^\star\|$ and the function value $f(x^k) - f^\star$ both converge nonmonotonically. The Lyapunov function found using the dissipativity approach is, however, monotone. The dissipativity interpretation is that some of the energy is stored in the function value, while some is stored in the states. Energy sloshes back and forth between both. However, the total energy is dissipated at a rate bounded by ρ^{2k}. In this case, the bound is loose; the algorithm converges significantly faster than its worst-case bound. In the second case (right panel of Figure 6), the energy is dissipated more slowly and matches the worst-case bound.

8 Case study: Alternating Direction Method of Multipliers

Consider the ADMM algorithm applied to a composite optimization problem $\min_x f(x) + g(x)$, as described by the interconnections shown in Figure 3. We investigate how to tune the stepsize η to

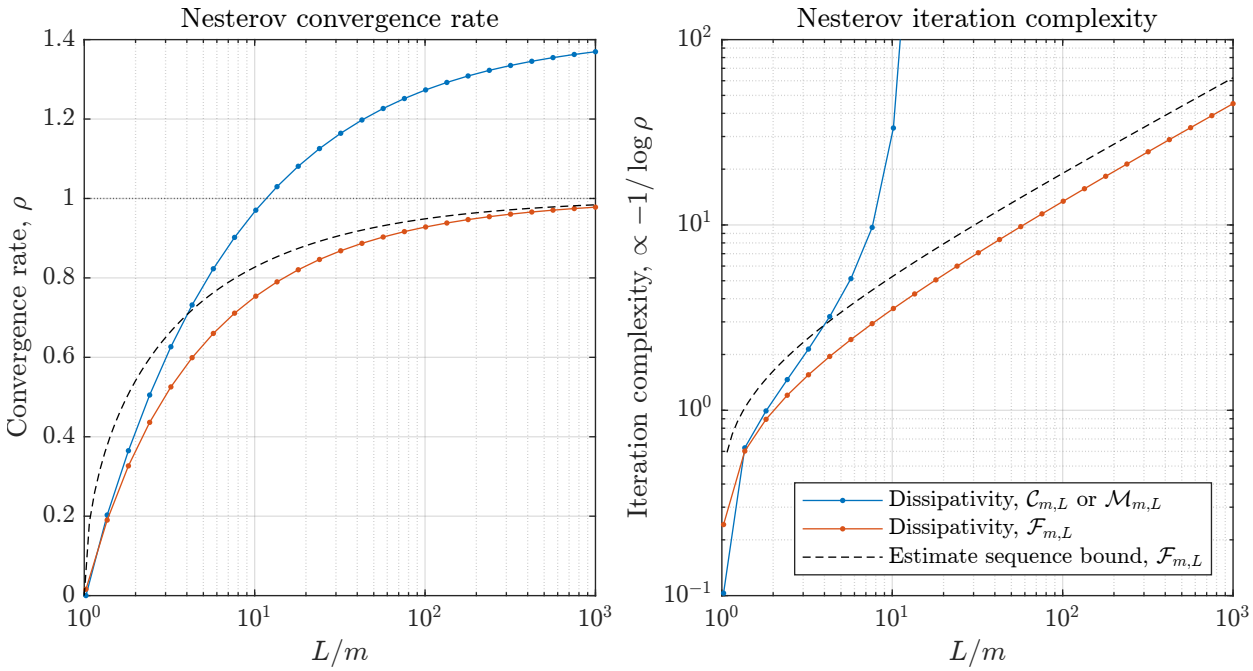


Figure 5. Worst-case convergence analysis of Nesterov's accelerated method with standard tuning. **Left:** Worst-case convergence rate ρ as a function of the condition number L/m. In the case where f is m-strongly convex with L-Lipschitz gradient ($\mathcal{F}_{m,L}$), the dissipativity approach yields an improvement on the bound found using estimate sequences [23, §2.2.1]. Also shown is the case where ∇f is only assumed to be sector-bounded ($\mathcal{C}_{m,L}$) or slope-restricted ($\mathcal{M}_{m,L}$), obtained using the dissipativity approach. Both cases have the same worst-case bound and are only guaranteed to converge ($\rho < 1$) when L/m is relatively small. **Right:** The same data as the left plot, but instead we plot $-1/\log\rho$, which is proportional to the iteration complexity (the number of iterations required to ensure convergence to within a prespecified tolerance). As $\rho \to 1$ on the left plot (slower convergence), the iteration complexity tends to $+\infty$ on the right plot. When $\rho > 1$, the algorithm may not converge, so iteration complexity is infinite. The dissipativity approach improves upon the estimate sequence bound by a constant factor of approximately 1.38.

ensure the fastest possible worst-case convergence. Assume f is strongly convex and g is convex. In other words, $\nabla f \in \mathcal{F}_{m,L}$ and $\partial g \in \mathcal{M}_{0,\infty}$. We can obtain an upper bound for the worst-case convergence rate using the supply rates for $\mathcal{C}_{m,L}$ and $\mathcal{C}_{0,\infty}$, respectively. Using the standard Lyapunov candidate

$$V(x) = x^{\mathsf{T}} P x, \quad \text{where } x := \begin{bmatrix} z \\ w \end{bmatrix} \text{ and } P \succ 0,$$

and a similar dissipation inequality to (20), which is used to analyze the gradient method:

$$V(x^{k+1}) - \rho^2 V(x^k) \leq \lambda_1 S_{\mathcal{C}}^{m,L}(y_1^k - y_1^\star, u_1^k) + \lambda_2 S_{\mathcal{C}}^{0,\infty}(y_2^k - y_2^\star, u_2^k). \tag{30}$$

In (30), we used superscripts with $S_{\mathcal{C}}$ to denote the bounds of the sector. For the case $S_{\mathcal{C}}^{0,\infty}$, which corresponds to the set $\mathcal{C}_{0,\infty}$, we divide (9) by L and take the limit $L \to \infty$, which yields

$$S_{\mathcal{C}}^{0,\infty}(y,u) = \begin{bmatrix} y \\ u \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 0 & -\frac{1}{2} \\ -\frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix}.$$

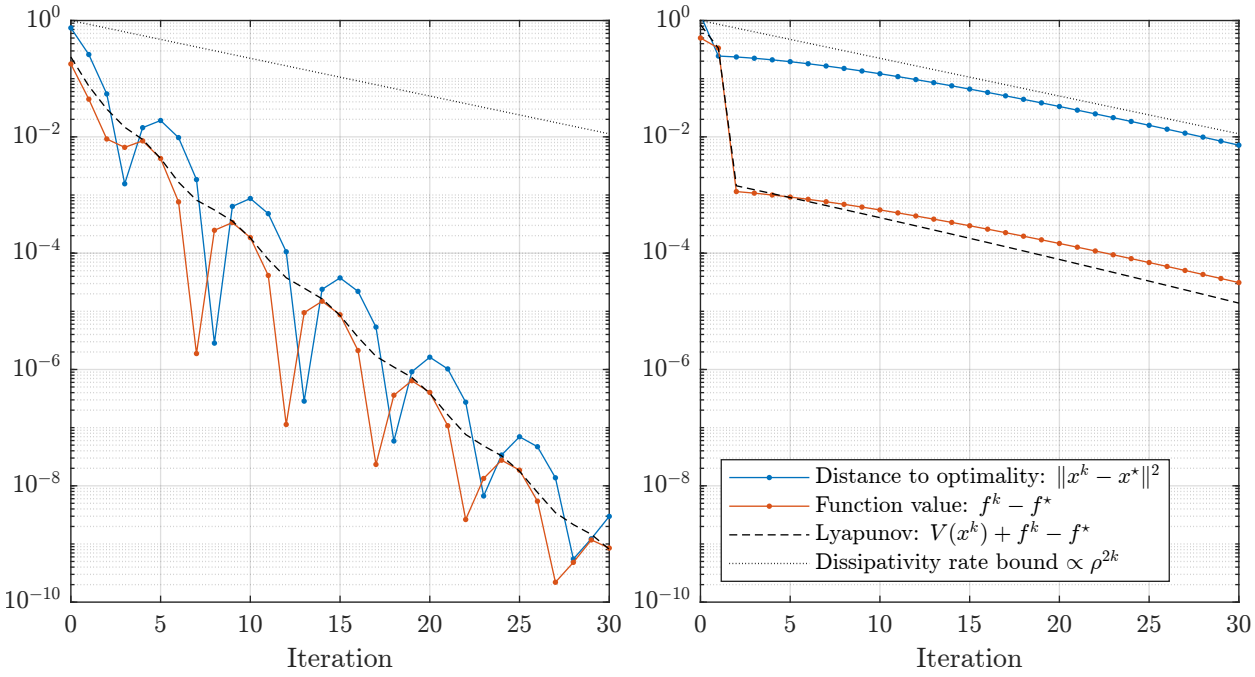


Figure 6. Simulation of Nesterov's accelerated method applied to two test functions that are m-strongly convex with L-Lipschitz gradients. In both cases, $L = 1$ and $m = \frac{1}{100}$ were used and the squared distance to optimality $\|x^k - x^\star\|^2$, the function value $f^k - f^\star$, and the Lyapunov function found using the dissipativity approach were plotted. **Left:** The function given in (29a). In this case, neither the distance to optimality nor the function value decrease monotonically. However, the Lyapunov function does. Convergence is faster than that predicted by the dissipativity approach. **Right:** The function given in (29b). In this case, the convergence is slower and the numerical bound from the dissipativity approach appears to be tight. For both functions, Nesterov's method was initialized with $x_1 = 1$ and $x_2 = 0.5$.

Upon substituting the dynamics (6) into the dissipation inequality (30), a quadratic inequality is obtained in the variables $\{z^k - z^\star, w^k - w^\star, u_1^k, u_2^k\}$, which yields the semidefinite inequality

$$\begin{aligned}\begin{bmatrix}\star\end{bmatrix}^\mathsf{T} P \begin{bmatrix}1 & 0 & -\eta & -\eta\\ 0 & 0 & 0 & \eta\end{bmatrix} - \rho^2 \begin{bmatrix}\star\end{bmatrix}^\mathsf{T} P \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}\\ \preceq \lambda_1 \begin{bmatrix}\star\end{bmatrix}^\mathsf{T} \begin{bmatrix}mL & -\frac{L+m}{2}\\ -\frac{L+m}{2} & 1\end{bmatrix} \begin{bmatrix}1 & -1 & -\eta & 0\\ 0 & 0 & 1 & 0\end{bmatrix} + \lambda_2 \begin{bmatrix}\star\end{bmatrix}^\mathsf{T} \begin{bmatrix}0 & -\frac{1}{2}\\ -\frac{1}{2} & 0\end{bmatrix} \begin{bmatrix}1 & 0 & -\eta & -\eta\\ 0 & 0 & 0 & 1\end{bmatrix},\end{aligned} \tag{31}$$

where $\star$ denotes the term required to make the quadratic forms symmetric. For example, in $\begin{bmatrix}\star\end{bmatrix}^\mathsf{T} PX$, $\star = X$. For each fixed ρ, η, m, L, (31) is an SDP in the variables $P \succ 0$ and $\lambda_1, \lambda_2 \geq 0$. For different choices of stepsize η and condition number L/m, (31) is solved using a bisection search to find the smallest ρ that ensures feasibility. Numerically obtained SDP upper bounds are plotted in Figure 7, alongside the analytical bound from [7, Cor. 3.1], which is given by

$$\rho_{\text{ub}} \leq \sqrt{\frac{\eta^2 Lm + 1}{\eta^2 Lm + 2\eta m + 1}}. \tag{32}$$

The convergence rate upper bounds shown in Figure 7 may be conservative because supply rates

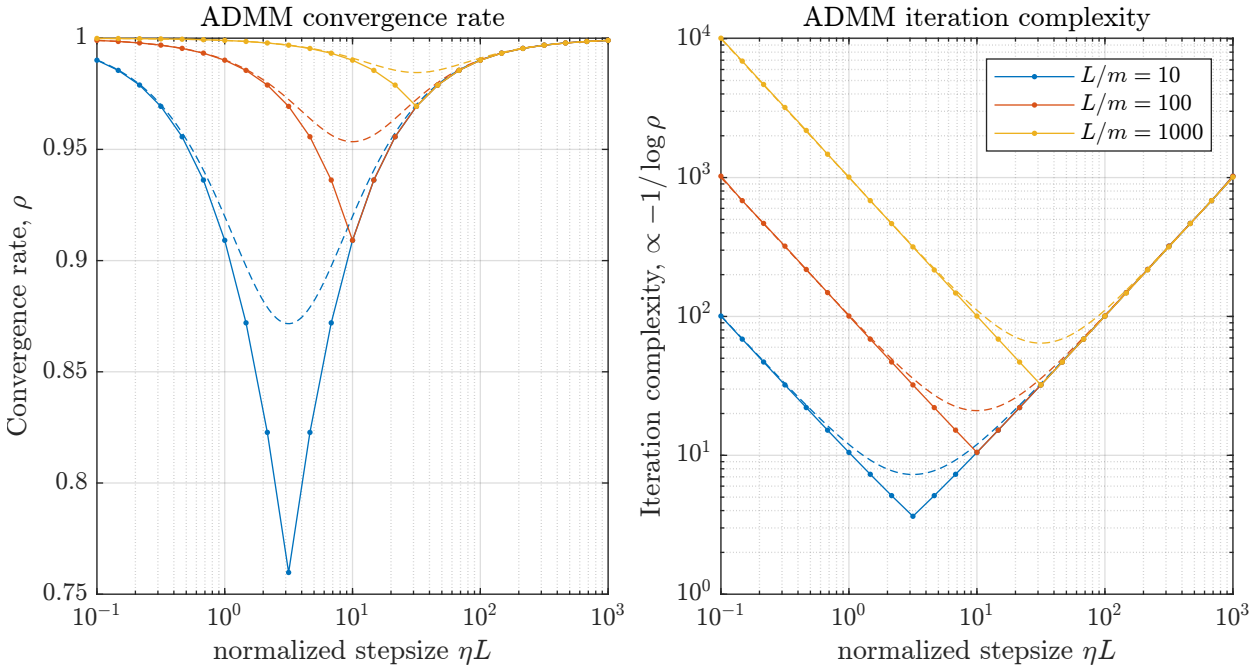


Figure 7. Worst-case convergence analysis of Alternating Direction Method of Multipliers (ADMM) applied to the problem of minimizing $f(x) + g(x)$, where f is m-strongly convex with L-Lipschitz gradient and g is convex but not necessarily differentiable. This includes the case of constrained optimization (where g is the indicator function of a convex set). Solid lines show the upper bound found by solving the dissipation inequality (31). Dashed lines show the upper bound (32) from [7, Cor. 3.1], which is looser than the SDP bound. **Left:** Worst-case convergence rate ρ as a function of the normalized stepsize ηL for different choices of L/m. **Right:** The same data as the left plot, but instead plot $-1/\log \rho$, which is proportional to the iteration complexity (the number of iterations required to ensure convergence to within a prespecified tolerance).

$\mathcal{C}_{m,L}$ and $\mathcal{C}_{0,\infty}$ were used rather than the more precise $\mathcal{F}_{m,L}$ and $\mathcal{M}_{0,\infty}$, respectively. Nevertheless, a matching lower bound can be found by considering a specific quadratic problem instance, as in [8, 24]. Consider the composite objective with $f(x) = \frac{1}{2}x^\mathsf{T}Qx$, where the largest and smallest eigenvalues of Q are given by L and m, respectively, and $g(x) = \frac{1}{2}\delta\|x\|^2$ with $\delta \geq 0$. In this case, (5) reduces to the linear update $z^{k+1} = (1+\eta\delta)^{-1}(I+\eta Q)^{-1}(I+\eta^2\delta Q)z^k$. If $\lambda \in [m, L]$ is an eigenvalue of Q, the eigenvalues of the update matrix for z^k are $(1+\eta\delta)^{-1}(1+\eta\lambda)^{-1}(1+\eta^2\delta\lambda)$. Therefore, the greatest lower bound is found by extremizing:

$$\rho_{\text{lb}} \geq \sup_{\delta\geq 0} \sup_{m\leq\lambda\leq L} \frac{1+\eta^2\delta\lambda}{(1+\eta\delta)(1+\eta\lambda)} \geq \max\left(\frac{1}{1+\eta m}, \frac{\eta L}{1+\eta L}\right),$$

where the second inequality is found by picking $\delta = 0$ and $\delta \to \infty$. This lower bound coincides precisely with the numerical SDP upper bound plotted in Figure 7, which suggests that the upper bound was tight. A rigorous proof that the upper and lower bounds match requires finding an analytic solution to (31). This SDP is larger and has more variables than the gradient descent SDP (21), which is why analytic solutions are more difficult to obtain in this case. By inspection, an analytic solution for the peaks of interest in Figure 7 can be found, which are the stepsizes η at

which the worst-case convergence rate ρ is fastest:

$$\eta = \frac{1}{\sqrt{Lm}}, \qquad \rho = \frac{\sqrt{L}}{\sqrt{L}+\sqrt{m}}, \qquad \lambda_1 = 1, \qquad \lambda_2 = \frac{(L-m)^2}{L},$$
$$P = \frac{\sqrt{m}(\sqrt{L}+\sqrt{m})(L-m)}{2}\begin{bmatrix} 1 & \sqrt{\frac{m}{L}} \\ \sqrt{\frac{m}{L}} & 1 \end{bmatrix}.$$

The SDP analysis above was carried out using the gradient oracles (6). The same worst-case rates are obtained if we instead used the proximal oracle formulation (5), provided suitable adjustments are made to the supply rates. For example, monotonicity of the subgradient ($\partial g \in \mathcal{M}_{0,\infty}$) corresponds to firm nonexpansiveness of the proximal operator ($\mathrm{prox}_{\eta g} \in \mathcal{M}_{0,1}$). Although the optimized convergence rate for ADMM, $\frac{\sqrt{L}}{\sqrt{L}+\sqrt{m}}$, is faster than that of gradient descent, $\frac{L-m}{L+m}$, the cost of a single iteration may be dramatically different in terms of wall-clock time, since computing one gradient is likely far cheaper than computing one proximal operation. Indeed, as $\eta \to \infty$, $\mathrm{prox}_{\eta f}(x) \to \arg\min_x f(x)$. So, when η is large, a single proximal operation solves the unconstrained optimization problem.

9 General scalable algorithm analysis

The approaches developed in the three previous case studies extend to algorithms in the general form (7). In other words, assume an algorithm with updates of the form

$$\begin{bmatrix} \xi^{k+1} \\ y_1^k \\ \vdots \\ y_m^k \end{bmatrix} = \begin{bmatrix} A & B_1 & \cdots & B_m \\ C_1 & D_{11} & \cdots & D_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ C_m & D_{m1} & \cdots & D_{mm} \end{bmatrix} \begin{bmatrix} \xi^k \\ u_1^k \\ \vdots \\ u_m^k \end{bmatrix} \tag{33}$$

and oracles $u_i^k = \phi_i(y_i^k)$ for $i = 1, \ldots, m$. Suppose we want to use a window of r consecutive timesteps in our analysis. Then, all relevant supply rates for each of the oracles are written as in the case study on Nesterov's method. These will depend on the pairs $(y^\star, u^\star), (y^k, u^k), \ldots, (y^{k+r}, u^{k+r})$. These supply rates are called $\{S^1, \ldots, S^M\}$. As before, choose a quadratic storage function $V(\xi^k, \ldots, \xi^{k+r})$. To simplify the exposition, consider the case where the oracles are sector-bounded or slope-restricted. This leads to the following inequality, similar to (25) and (26):

$$V(\xi^{k+1}, \ldots, \xi^{k+r+1}) - \rho^2 V(\xi^k, \ldots, \xi^{k+r}) \le \sum_{i=1}^{M} \lambda_i S^i, \tag{34}$$

where the λ_i are nonnegative. If some of the oracles are gradients of strongly convex functions instead, we also include the positivity inequality as in (27b) and the associated linear constraints on the λ_i and μ_i. The supply rates are quadratic functions of pairs of inputs and outputs, and upon substituting the dynamics (33) to eliminate $\xi^{k+1}, \ldots, \xi^{k+r+1}$. The inequality (34) reduces to a quadratic expression in the variables $\{\xi^k, u^k, \ldots, u^{k+r}\}$, where $u^{k+i} := (u_1^{k+i}, \ldots, u_m^{k+i})$. How large is the semidefinite program associated with (34)? If it is assumed that the algorithm has n states and the oracles each map $\phi_i : \mathbb{R}^d \to \mathbb{R}^d$, then $A \in \mathbb{R}^{nd \times nd}$, $B_i \in \mathbb{R}^{nd \times d}$, $C_i \in \mathbb{R}^{d \times nd}$, and $D_{ij} \in \mathbb{R}^{d \times d}$. The storage function depends on $\xi^k, \ldots, \xi^{k+1}$, so the quadratic form can be represented using a

symmetric matrix $P \in \mathbb{R}^{n(r+1)d \times n(r+1)d}$. Note also the scalar variables $\lambda_1, \ldots, \lambda_M$. Meanwhile, the entire inequality (34) depends on $\{\xi^k, u^k, \ldots, u^{k+r}\}$, so it is a semidefinite constraint of size $(n+r+1)d \times (n+r+1)d$.

9.1 Scalability

For typical iterative algorithms, n is small; $n = 1$ for gradient descent and $n = 2$ for Nesterov's method. Tight convergence guarantees can also be obtained with relatively small r; our case studies required $r = 1$ or $r = 2$. However, in machine learning applications, it is not uncommon for d to be very large (millions or billions), which would make the semidefinite program described above prohibitively large. Although d may be large, algorithms typically have highly structured system matrices (A, B, C, D). For example, it is shown from (17) that gradient descent has diagonal system matrices:

$$A = I_d, \qquad B = -\eta I_d, \qquad C = I_d, \qquad D = 0,$$

where $I_d \in \mathbb{R}^{d \times d}$ is the identity matrix. Similarly, it is apparent from (23) that Nesterov's method has block-diagonal system matrices:

$$A = \begin{bmatrix} 1+\beta & -\beta \\ 1 & 0 \end{bmatrix} \otimes I_d, \qquad B = \begin{bmatrix} -\eta \\ 0 \end{bmatrix} \otimes I_d, \qquad C = \begin{bmatrix} 1+\beta & -\beta \end{bmatrix} \otimes I_d, \qquad D = 0,$$

where $\otimes$ denotes the Kronecker product. A similar structure is present in the supply rates. Therefore, the P matrix from the storage function may also be written as $P = \hat{P} \otimes I_d$, where $\hat{P} \in \mathbb{R}^{n(r+1) \times n(r+1)}$. Thus, the semidefinite program decouples into d identical (and much smaller) semidefinite programs, meaning that the dissipativity-based worst-case algorithm analysis only requires solving small semidefinite programs whose size depends on n and r, but not d.

10 Concluding remarks

The case studies above show how dissipativity can be applied to the analysis of iterative optimization algorithms. Further examples can be analyzed in an analogous manner, including distributed optimization algorithms [29, 31], stochastic and variance-reduction algorithms [12–14], and alternating algorithms for smooth games [41]. The benefit of using dissipativity for algorithm analysis is that it provides a principled and modular framework where algorithms and oracles can be interchanged and analyzed depending on the case at hand. Additionally, the dissipativity approach is computationally tractable. The semidefinite programs solved in the case studies are small, with fewer than 20 variables. Importantly, the size of the semidefinite programs is independent of the dimension of the domain of the objective function.

11 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. 1656951, 1750162, 1936648.

References

[1] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces, Second edition*. Springer, 2017.

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

[3] J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Mathematical Programming*, 165(2):471–507, 2017.

[4] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

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

[7] W. Deng and W. Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66(3):889–916, 2016.

[8] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

[9] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, Mar. 2014.

[10] W. P. Heath and A. G. Wills. Zames-Falb multipliers for quadratic programming. In *IEEE Conference on Decision and Control*, pages 963–968, 2005.

[11] B. Hu and L. Lessard. Dissipativity theory for Nesterov's accelerated method. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1549–1557, 2017.

[12] B. Hu, P. Seiler, and L. Lessard. Analysis of biased stochastic gradient descent using sequential semidefinite programs. *Mathematical Programming*, Mar. 2020.

[13] B. Hu, P. Seiler, and A. Rantzer. A unified analysis of stochastic optimization methods using jump system theory and quadratic constraints. In *Conference on Learning Theory*, pages 1157–1189. PMLR, 2017.

[14] B. Hu, S. Wright, and L. Lessard. Dissipativity theory for accelerating stochastic variance reduction: A unified analysis of SVRG and Katyusha using semidefinite programs. In *International Conference on Machine Learning*, pages 2038–2047, July 2018.

[15] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[16] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[17] H. K. Khalil and J. W. Grizzle. *Nonlinear systems*, volume 3. Prentice Hall Upper Saddle River, NJ, 2002.

[18] 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.

[19] A. I. Lur'e and V. N. Postnikov. On the theory of stability of control systems. *Applied mathematics and mechanics*, 8(3):246–248, 1944. In Russian.

[20] A. M. Lyapunov and A. T. Fuller. *General Problem of the Stability Of Motion*. Control Theory and Applications Series. Taylor & Francis, 1992. Original text in Russian, 1892.

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

[22] S. Michalowsky, C. Scherer, and C. Ebenbauer. Robust and structure exploiting optimisation algorithms: an integral quadratic constraint approach. *International Journal of Control*, pages 1–24, 2020.

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

[24] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. Jordan. A general analysis of the convergence of ADMM. In *International Conference on Machine Learning*, pages 343–352. PMLR, 2015.

[25] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[26] B. T. Polyak. *Introduction to optimization*. Optimization Software, Inc., New York, 1987.

[27] R. T. Rockafellar. *Convex analysis*. Princeton University Press, 2015.

[28] S. Safavi, B. Joshi, G. França, and J. Bento. An explicit convergence rate for nesterov's method from sdp. In *2018 IEEE International Symposium on Information Theory*, pages 1560–1564, 2018.

[29] B. V. Scoy and L. Lessard. Systematic analysis of distributed optimization algorithms over jointly-connected networks. In *IEEE Conference on Decision and Control*, pages 3096–3101, Dec. 2020.

[30] P. Seiler. Stability analysis with dissipation inequalities and integral quadratic constraints. *IEEE Transactions on Automatic Control*, 60(6):1704–1709, 2015.

[31] A. Sundararajan, B. V. Scoy, and L. Lessard. Analysis and design of first-order distributed optimization algorithms over time-varying graphs. *IEEE Transactions on Control of Network Systems*, 7(4):1597–1608, Dec. 2020.

[32] 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(1-2):307–345, 2017.

[33] Y. Z. Tsypkin and Z. J. Nikolic. *Adaptation and learning in automatic systems*, volume 73. Academic Press New York, 1971.

[34] B. Van Scoy, R. A. Freeman, and K. M. Lynch. The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Systems Letters*, 2(1):49–54, 2017.

[35] J. Veenman, C. W. Scherer, and H. Köroğlu. Robust stability and performance analysis based on integral quadratic constraints. *European Journal of Control*, 31:1–32, 2016.

[36] J. C. Willems. Dissipative dynamical systems part I: General theory. *Archive for rational mechanics and analysis*, 45(5):321–351, 1972.

[37] J. C. Willems. Dissipative dynamical systems part II: Linear systems with quadratic supply rates. *Archive for rational mechanics and analysis*, 45(5):352–393, 1972.

[38] G. Zames. On the input-output stability of time-varying nonlinear feedback systems—Part I: Conditions derived using concepts of loop gain, conicity, and positivity, and Part II: Conditions involving circles in the frequency plane and sector nonlinearities. *IEEE Transactions on Automatic Control*, 11(2,3):228–238,465–476, 1966.

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

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

[41] G. Zhang, X. Bao, L. Lessard, and R. Grosse. A unified analysis of first-order methods for smooth games via integral quadratic constraints. *Journal of Machine Learning Research*, 22(103):1–39, 2021.

A 50 years of dissipativity in algorithm analysis

Lyapunov theory [20] is the main tool for the stability analysis of dynamical systems and has become a cornerstone of control theory as well. Optimization algorithms have a natural interpretation as dynamical systems and therefore can also be analyzed using Lyapunov theory. The idea of viewing iterative algorithms as dynamical systems started perhaps in the early 1970s with the work of Tsypkin [33]. Tsypkin draws many parallels between optimization algorithms and control systems, with an emphasis on gradient-based algorithms. The main connections are outlined in the table below.

Optimization	**Controls**
algorithm converges to a minimizer	equilibrium point is asymptotically stable
algorithm converges to a minimizer for all functions in a given class	robust stability
designing an algorithm with bounds on its worst-case performance	robust controller synthesis

At around the same time, Willems published his seminal work on dissipativity [36, 37], which generalized Lyapunov theory to systems with inputs, and he also generalized previous robust stability criteria such as passivity theory and the small-gain theorem [38]. These ideas are at the core of nonlinear systems theory and endure to this day [17]. Another key body of work is Polyak's *Introduction to Optimization* [26], which covers the fundamentals of iterative methods for continuous and convex optimization. Polyak's book adopts a dynamical systems perspective, and even features a chapter on Lyapunov's method. The problem of worst-case algorithm analysis is fundamentally a Lur'e problem [19]. An important distinction is that in the robust control literature, the goal is typically to prove *stability*, whereas in algorithm analysis, the goal is to quantify the rate of convergence (which is akin to controlling the rate of exponential convergence). The modern tool for tackling this problem is integral quadratic constraints (IQCs), either in the frequency domain [21,35] or in the time domain via dissipativity [30]. More recent works study algorithm analysis through the dynamical systems viewpoint have used IQC and dissipativity tools to achieve the tightest known bounds on many popular algorithms [11, 18, 24]. These ideas have also been extended to synthesis, in an effort to design new optimization algorithms in a principled way [6, 22, 31, 34] Although this survey focused mostly on gradient-based iterative methods for continuous optimization, control perspectives have also been applied to (1) iterative methods for solving linear systems, (2) algorithms for solving ordinary differential equations, and (3) algorithms for solving linear programs. For a comprehensive survey of these topics, see [2].

B Proving inequalities via the S-procedure

In the optimization literature, inequalities such as (10) are typically proved on a case-by-case basis using clever combinations of the triangle inequality, Cauchy–Schwarz, and other inequalities. Instead, the more fundamental property of positivity can often be used, namely if $S \leq 0$ is known to be true and $S = Q + P$ is split, where $P \geq 0$, then it follows that $Q \leq 0$. Start with the definition

$$S_{\mathcal{C}}(y,u) := \begin{bmatrix} y \\ u \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} mL & -\frac{L+m}{2} \\ -\frac{L+m}{2} & 1 \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix}$$

and consider the algebraic identity

$$S_{\mathcal{C}}(y,u) = \tfrac{L-m}{2m}\big(m^2\|y\|^2 - \|u\|^2\big) + \tfrac{L+m}{2m}\|my-u\|^2, \tag{35a}$$

which can be directly verified to hold for all y and u by expanding both sides and comparing like terms. Since $0 < m \leq L$ by assumption, the second term in (35a) is nonnegative, as norms are always nonnegative. Therefore, by the positivity property, if $S_{\mathcal{C}}(y,u) \leq 0$, then $m^2\|y\|^2 - \|u\|^2 \leq 0$. In other words, $m\|y\| \leq \|u\|$, which is the first half of (10). The remaining inequalities in (10)–(11) can be proven using a similar approach, with the further algebraic identities:

$$S_{\mathcal{C}}(y,u) = \tfrac{L-m}{2L}\big(\|u\|^2 - L^2\|y\|^2\big) + \tfrac{L+m}{2L}\|u-Ly\|^2, \tag{35b}$$

$$S_{\mathcal{C}}(y,u) = (L-m)\big(m\|y\|^2 - u^{\mathsf{T}}y\big) + \|my-u\|^2, \tag{35c}$$

$$S_{\mathcal{C}}(y,u) = (L-m)\big(u^{\mathsf{T}}y - L\|y\|^2\big) + \|u-Ly\|^2. \tag{35d}$$

This approach of proving that certain quadratic inequalities hold when others do is closely related to the *S-procedure* from control theory [4, pp. 23, 33]. The lossless S-procedure states that the two following statements are equivalent:

1. For all x, if $x^{\mathsf{T}}Sx \leq 0$, then $x^{\mathsf{T}}Qx \leq 0$.
2. There exists $\lambda \geq 0$ such that $S \succeq \lambda Q$.

The S-procedure allows inequalities such as (35) to be generated systematically. For example, to generate (35a), seek a $\lambda \geq 0$ such that

$$S_{\mathcal{C}}(y,u) \geq \lambda\left(m^2\|y\| - \|u\|^2\right) \quad \text{for all } y,u \tag{36}$$

The inequality (36) is equivalent to the semidefinite program:

$$\text{Find } \lambda \geq 0 \text{ such that: } \begin{bmatrix} mL & -\frac{L+m}{2} \\ -\frac{L+m}{2} & 1 \end{bmatrix} \succeq \lambda \begin{bmatrix} m^2 & 0 \\ 0 & -1 \end{bmatrix}. \tag{37}$$

It can be shown that the unique solution to (37) is $\lambda = \frac{L-m}{2m}$, which leads to (35a). Since the S-procedure is *lossless* (necessary and sufficient), it will always find an algebraic identity if one exists. In other words, if there is no λ that satisfies (37), then $S_{\mathcal{C}}(y,u) \leq 0$ *does not* imply that $m\|y\| \leq \|u\|$ for all y,u.