Analysis and Design of First-Order Distributed Optimization Algorithms over Time-Varying Graphs

Akhil Sundararajan[1,2] Bryan Van Scoy[1] Laurent Lessard[1,2]

Abstract

This work concerns the analysis and design of distributed first-order optimization algorithms over time-varying graphs. The goal of such algorithms is to optimize a global function that is the average of local functions using only local computations and communications. Several different algorithms have been proposed that achieve linear convergence to the global optimum when the local functions are strongly convex. We provide a unified analysis that yields a worst-case linear convergence rate as a function of the condition number of the local functions, the spectral gap of the graph, and the parameters of the algorithm. The framework requires solving a small semidefinite program whose size is fixed; it does not depend on the number of local functions or the dimension of the domain. The result is a computationally efficient method for distributed algorithm analysis that enables the rapid comparison, selection, and tuning of algorithms. Finally, we propose a new algorithm, which we call SVL, that is easily implementable and achieves the fastest possible worst-case convergence rate among all algorithms in the family we considered. We support our theoretical analysis with numerical experiments that generate worst-case examples demonstrating the effectiveness of SVL.

1 Introduction

In distributed optimization, a network of agents (such as computing nodes, robots, or mobile sensors) work collaboratively to optimize a global objective. Specifically, each agent $i \in \{1, \dots, n\}$ has access to a local function f_i and must minimize the average of all the local functions:

$$\min_{x \in \mathbb{R}^d} f(x), \quad \text{where } f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x). \tag{1}$$

Distributed optimization has been a topic of interest in recent years as it arises in many applications. For example, in large-scale machine learning, n could represent the number of computing units available for training a large data set. Each f_i then denotes a loss function corresponding to the subset of training examples assigned to that particular unit, and the global objective is to solve the aggregate problem that uses all agents' data. Another example is a sensor network that collects data in a distributed way. Each individual sensor may have a limited power budget, communication bandwidth, or sensing capability. It is desirable to accomplish a global task, such as aggregating all locally collected data, without placing a great burden on any individual sensing node or having

[1]Wisconsin Institute for Discovery, WI 53715, USA.
[2]Department of Electrical and Computer Engineering, University of Wisconsin–Madison, WI 53706, USA.
Emails: {asundararaja,vanscoy,laurent.lessard}@wisc.edu

a single point of failure. Other application areas include optimization in sensor networks [22]; distributed spectrum sensing [2]; resource allocation across geographic regions [23]; and large-scale learning tasks, where the aim is to speed up training via multiple processors [8, 10].

A popular abstraction is to imagine the agents as nodes on a graph where the edges indicate which inter-node communications are permitted. Several iterative methods that solve (1) have been proposed in which each agent will perform some combination of local gradient computations, local memory updates, and local averaging of information with its neighbors.

If no optimization is required, we have the special case of *average consensus* [30, 31]. This amounts to each agent i using the starting value x_i^0 and local objective $f_i(x) = \|x - x_i^0\|^2$. Then, the unique optimizer of (1) is $x^\star = \frac{1}{n}\sum_{i=1}^n x_i^0$, the average value of all initial local states. Simple local averaging, also known as *gossip*, uses updates of the form: $x_i^{k+1} = \sum_{i=1}^n W_{ij} x_j^k$ where W is a carefully chosen matrix with the same sparsity pattern as the adjacency matrix of the communication graph. Such methods achieve a linear rate of convergence in the sense that $\|x_i^k - x^\star\| \leq \rho^k$ for some rate parameter $0 < \rho < 1$ that depends on W [32]. On the other hand, if no communication is required, which happens when $n = 1$ or if all the f_i are identical, we recover the standard centralized optimization setup. Depending on the properties of f_i, it may also be possible to guarantee a linear convergence rate in this setting. For example, simple gradient descent iterations of the form $x_i^{k+1} = x_i^k - \alpha \nabla f_i(x_i^k)$ achieve a linear rate of convergence when f_i is continuously differentiable, smooth, and strongly convex (formally stated in Assumption 1) [18].

Solving (1) in the general case involves a combination of gossip and optimization. However, it is not straightforward to design algorithms that obtain linear convergence rates. An early attempt is *distributed gradient descent* (DGD) [17], which uses the update rule

$$x_i^{k+1} = \sum_{i=1}^n W_{ij}\, x_j^k - \alpha\, \nabla f_i(x_i^k) \tag{2}$$

and is a straightforward combination of gossip and gradient descent. In general, DGD requires a diminishing stepsize α in order to converge to the optimal solution of (1), and convergence happens at a sublinear rate even when the f_i are smooth and strongly convex. The intuition behind this fact is that the optimal point $x^\star$ is not necessarily a fixed point of the algorithm since it is in general not a minimizer of each individual f_i.

A linear convergence rate for the general case was first achieved by the *exact first-order algorithm* (EXTRA) [26]. This algorithm requires that the previous state be stored in memory and uses the iteration

$$x_i^1 = \sum_{j=1}^n W_{ij}\, x_j^0 - \alpha\, \nabla f_i(x_i^0), \qquad x_i^0 \text{ arbitrary}, \tag{3a}$$

$$x_i^{k+2} = x_i^{k+1} + \sum_{j=1}^n W_{ij}\, x_j^{k+1} - \sum_{j=1}^n \widetilde{W}_{ij}\, x_j^k - \alpha \left(\nabla f_i(x_i^{k+1}) - \nabla f_i(x_i^k)\right) \tag{3b}$$

where W and $\widetilde{W}$ are gossip matrices that satisfy certain technical conditions and α is sufficiently small. Although the update (3b) has two gradient evaluations, they are evaluated at successive iterates so EXTRA can easily be implemented with one gradient evaluation per iteration by storing the previous gradient in memory. Several additional linear-rate algorithms have since been proposed [9,13,16,21,36,38,39]. Each of these methods have updates similar to (3) in that they require

agents to store the previous iterate and/or gradient in memory. Other distributed optimization algorithms solve (1) but lie outside the scope of the present work. This includes algorithms involving dual decomposition [4,24,25], inexact dual methods [7], proximal algorithms [27], asynchronous algorithms [15,37], weakly convex cases [13,20,26], accelerated methods [20,33,34].

Although linear convergence rates were obtained for many of the algorithms cited above, each algorithm differs in the nature and strength of its convergence analysis guarantees. For example, some works show (non-constructively) the existence of a linear rate [35] whereas others provide specific tuning recommendations with associated analytic rate bounds [13,26]. Numerical simulations are another means for comparing performance [33], but they run the risk of being misleading because algorithm performance often depends on the graph topology, choice of functions, hyperparameter tuning, or state initializations.

The present work marks an effort to systematize the analysis and design of distributed optimization algorithms. We center our attention on first-order methods involving gossip and study their performance on smooth strongly convex functions and time-varying graphs. We present a standardized framework for rate-of-convergence analysis of algorithms in this class. Our method also leads to the synthesis of a new algorithm with strong theoretical guarantees that we support with numerical simulations. We now summarize our main contributions.

- **Analysis framework.** We present a universal analysis framework that characterizes the worst-case linear convergence rate ρ of a wide range of distributed algorithms as a function of the parameters κ (a measure of how well-conditioned the local functions are) and σ (a measure of how well-connected the network is). Our main result is a semidefinite program (SDP) parameterized by κ, σ, and ρ, whose solution yields an upper bound on ρ. The SDP has a small fixed size that does not depend on the number of agents n or the dimension of the function domains. Our SDP yields robust performance guarantees when the graph is allowed to change at each iteration.

- **Algorithm design.** Our second contribution is a new distributed algorithm, which we name SVL (the authors' initials). Our algorithm is derived by optimizing the SDP from our analysis framework and provides the fastest known convergence rate to date. SVL depends explicitly on κ and σ, so no tuning is required if these parameters are known or estimated in advance. Figure 1 compares the worst-case linear rate ρ for several algorithms that we can certify using our analysis framework. We compare different tunings of EXTRA [26] and NIDS [13] along with our proposed algorithm, SVL. When σ is small (the graph is well-connected), SVL performs as well as ordinary centralized gradient descent, but when σ gets sufficiently large (the graph is poorly connected), the worst-case linear rate slows down.

- **Worst-case examples.** Although our analysis technique only provides upper bounds on the worst-case convergence rate of a distributed algorithm, we outline a computationally tractable optimization procedure that finds numerically matching lower bounds, which suggests the bounds found via our SDP are tight.

The paper is organized as follows. We lay out our assumptions about the functions, graph, and class of algorithms used in our framework in Section 2. We state and prove our main SDP result for certifying worst-case linear rate bounds in Section 3. We present our SVL algorithm and discuss interpretations in Section 4. Finally, we demonstrate the tightness of our bounds by generating worst-case algorithm trajectories and comparing SVL with NIDS and EXTRA in Section 5.

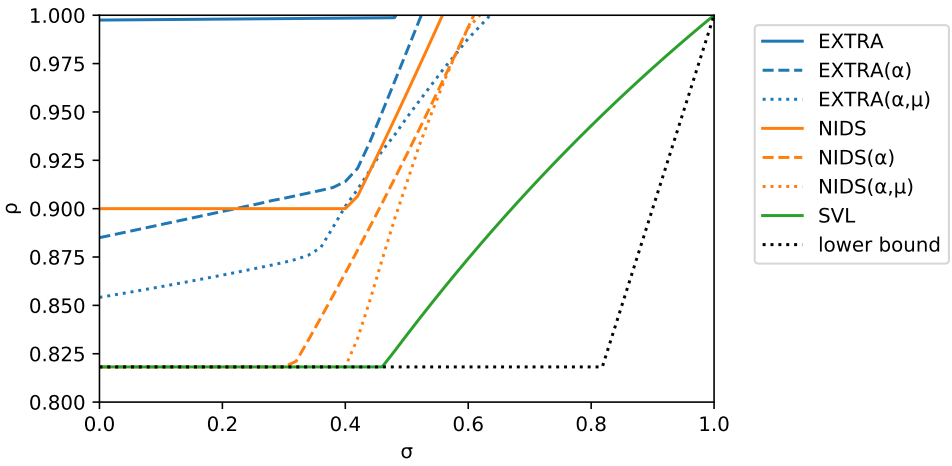

Figure 1: Worst-case convergence rate ρ (smaller is better) certified for the distributed algorithms NIDS, EXTRA, and our proposed algorithm, SVL, for a condition ratio $\kappa = 10$. A value of $\sigma = 0$ corresponds to a fully connected graph while $\sigma = 1$ corresponds to a completely disconnected graph. When $\rho \geq 1$, we can no longer certify that the algorithm converges in the worst case.

2 Preliminaries

2.1 Notation

Let I_p be the identity matrix in $\mathbb{R}^{p\times p}$. The symbols 1_n and 0_n denote the column vectors of all ones and all zeros in $\mathbb{R}^n$, respectively. The boldface symbol $\mathbf{0}_n$ denotes the $n \times n$ zero matrix. $\Pi_n := \frac{1}{n} 1_n 1_n^\mathsf{T}$ is the projection matrix onto 1_n. In these cases, we will sometimes omit subscripts when dimensions are clear from context. Unless otherwise indicated, Greek letters denote scalar parameters, lower-case letters denote column vectors, and upper-case letters denote matrices. Exceptions include the scalar parameters m and L, which we used in Assumption 1 to conform with conventional notation. The symbol $\otimes$ denotes the Kronecker matrix product. $\|x\|$ denotes the standard Euclidean norm of a vector x, and $\|A\| := \sup_{x\neq 0} \|Ax\|/\|x\|$ is the spectral norm of a matrix A. Throughout this paper, variables u, v, w, x, y are vectors. Unless otherwise indicated, subscripts always refer to individual agents while superscripts refer to time or iteration count.

Define the graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} := \{1, \ldots, n\}$ is the set of agents and $\mathcal{E}$ is the set of pairs of agents (i, j) that are connected. $\mathcal{L} \in \mathbb{R}^{n\times n}$ is a *Laplacian matrix* associated with $\mathcal{G}$ if $\mathcal{L}1_n = 0_n$ and $\mathcal{L}_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. The *spectral gap* of $\mathcal{L}$ is defined as the second-smallest eigenvalue magnitude of $\mathcal{L}$. Since we consider time-varying graphs, we let $\mathcal{L}^k$ denote a Laplacian of the graph $\mathcal{G}^k$.

2.2 Function and Graph Assumptions

We assume that each local function is strongly convex and has Lipschitz continuous gradients. This is described in the following assumption.

Assumption 1. *Given $0 < m \leq L$, the local functions f_i are continuously differentiable and satisfy*

$$m\|x - y\|^2 \leq \big(\nabla f_i(x) - \nabla f_i(y)\big)^\mathsf{T}(x - y) \leq L\|x - y\|^2 \qquad \text{for all } x, y \in \mathbb{R}^d \text{ and all } i \in \{1, \ldots, n\}.$$

We define the condition ratio as $\kappa := L/m$. This quantity captures a notion of how much the curvature of the objective function varies. In the case where f is twice differentiable, κ is an upper bound on the condition number of the Hessian $\nabla^2 f$. When $\kappa = 1$, each f_i is quadratic and relatively easy to optimize. As $\kappa \to \infty$, the functions become poorly conditioned and more difficult to optimize.

The graph associated with the network of agents can change at each step of the algorithm, so we make the following assumptions about the sequence of graph Laplacian matrices $\{\mathcal{L}^k\}$.

Assumption 2. *The following properties hold at each step of the algorithm.*

1. *The graph is connected, so there always exists a path between any two nodes in $\mathcal{G}^k$. This implies that the zero eigenvalue of $\mathcal{L}^k$ has a multiplicity of one for all k.*

2. *The graph is balanced, so the in-degree is equal to the out-degree of every node. This means that $1_n^\mathsf{T}\mathcal{L}^k = 0_n^\mathsf{T}$ for all k.*

3. *The spectral gap of the time-varying graph is uniformly bounded. In particular, we will assume there exists some $\sigma \in [0, 1)$ such that $\|I_n - \Pi_n - \mathcal{L}^k\| \leq \sigma$ for all k. Since the spectral radius of a matrix is always upper-bounded by its spectral norm, this implies that σ is a uniform bound on the spectral gap of each Laplacian matrix in $\{\mathcal{L}^k\}$.*

Assumptions 1 and 2 lead to quadratic inequalities that will be useful in proving our main result. These are stated in the following propositions.

Proposition 1. *Suppose Assumption 1 holds for the local objective functions f_i. For any $y^k, y^\ell \in \mathbb{R}^d$ define $u_i^k := \nabla f_i(y^k)$ and $u_i^\ell := \nabla f(y^\ell)$. Then*

$$\begin{bmatrix} y^k - y^\ell \\ u^k - u^\ell \end{bmatrix}^\mathsf{T} \left(\begin{bmatrix} -2mL & L+m \\ L+m & -2 \end{bmatrix} \otimes I_d \right) \begin{bmatrix} y^k - y^\ell \\ u^k - u^\ell \end{bmatrix} \geq 0.$$

Proof. This quadratic inequality follows from the definitions of smoothness and strong convexity and the co-coercivity property. A proof may be found in [18, Thm. 2.1.12]. ∎

Proposition 2. *Suppose Assumption 2 holds for the graph $\mathcal{G}^k$ at each iteration. For any $x^k \in \mathbb{R}^{nd}$ and $x^\star \in \mathbb{R}^d$, define $v^k := (\mathcal{L}^k \otimes I_d)x^k$. Then*

$$\begin{bmatrix} x^k - (1_n \otimes I_d)x^\star \\ v^k \end{bmatrix}^\mathsf{T} \left(\begin{bmatrix} \sigma^2 - 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes (I - \Pi) \otimes I_d \right) \begin{bmatrix} x^k - (1_n \otimes I_d)x^\star \\ v^k \end{bmatrix} \geq 0.$$

Proof. See Appendix A.1. ∎

2.3 Canonical Form

In this paper, we consider distributed algorithms that belong to the a canonical form parameterized by four scalars $(\alpha, \beta, \gamma, \delta)$. The generic algorithm is described in Algorithm 1 below.

Algorithm 1 (canonical form)

Initialization: Let $\mathcal{L}^k \in \mathbb{R}^{n\times n}$ be a Laplacian matrix. Each agent $i \in \{1,\dots,n\}$ chooses initial local state memory $x_i^0 \in \mathbb{R}^d$ arbitrarily and $w_i^0 \in \mathbb{R}^d$ such that $\sum_{i=1}^n w_i = 0$.

for iteration $k = 0, 1, 2, \dots$ **do**

  for agent $i \in \{1,\dots,n\}$ **do**

    Local communication

$$v_i^k = \sum_{j=1}^n \mathcal{L}_{ij}^k \, x_j^k \tag{C.1}$$

    Local gradient computation

$$y_i^k = x_i^k - \delta \, v_i^k \tag{C.2}$$

$$u_i^k = \nabla f_i(y_i^k) \tag{C.3}$$

    Local state update

$$x_i^{k+1} = x_i^k + \beta \, w_i^k - \alpha \, u_i^k - \gamma \, v_i^k \tag{C.4}$$

$$w_i^{k+1} = w_i^k - v_i^k \tag{C.5}$$

  end for

end for

This algorithmic template is derived from the slightly more general canonical form presented in [29]. Specifically, the five parameters used in [29] map to our four parameters via $(\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3) \mapsto (\alpha, \beta, \gamma, 0, \delta)$. The parameter ζ_2 mediates the communication of a second variable by each agent, which is required to represent algorithms such as AsynDGM [37] and DIGing [16]. However, such algorithms are not guaranteed to have a fixed point when the graph Laplacian $\mathcal{L}^k$ is time-varying. We set $\zeta_2 = 0$ in this paper to guarantee that all algorithms we consider have a fixed point. Specifically, instances of Algorithm 1 satisfy the following existence and uniqueness properties, which we prove in the Appendix.

Proposition 3. *Consider an algorithm with local state dimension at most $2d$ where each iteration consists of: simultaneously communicating any number of local variables with immediate neighbors, computing a local gradient evaluated at a computable point, and updating the local state. These three steps may be performed in any order. Further, assume the updates are linear, time-invariant, deterministic, and homogeneous across all agents and dimensions of the objective function. Finally, assume the algorithm converges to a unique fixed point that is the same for all choices of Laplacian sequences $\{\mathcal{L}^k\}$ satisfying Assumption 2. Then, any algorithm satisfying the aforementioned assumptions can be written in the form of Algorithm 1 for some choice of $(\alpha, \beta, \gamma, \delta)$.*

Conversely, consider an instance of Algorithm 1 with $\alpha, \beta \neq 0$. This algorithm has a fixed point that is the same for all choices of Laplacian sequences $\{\mathcal{L}^k\}$ satisfying Assumption 2. Moreover, this fixed point achieves consensus in the x-variables; $x_i^\star = x^\star$ where $x^\star$ satisfies $\nabla f(x^\star) = 0$.

Remark 4. *The second statement in Proposition 3 guarantees that any instance of Algorithm 1 has a desirable fixed point in the presence of a time-varying graph; all agents are in consensus and they agree on a stationary point of* (1)*. However, Proposition 3 does not ensure that Algorithm 1 necessarily converges to this fixed point, nor does it characterize the rate of convergence. These questions will be explored in Section 3.*

Two known distributed algorithms fit into the form of Algorithm 1: EXTRA [26] and NIDS [13], and we compare them in Table 1. In these algorithms, the stepsize parameter α is tunable and may need to be changed depending on the problem instance. We included the authors' recommended

α tuning in the table. Beyond the stepsize parameter, there is another way these algorithms can be tuned. Akin to the method of successive overrelaxation used in the numerical solutions of linear systems of equations [19], we may introduce a parameter μ that scales the Laplacian matrix. Table 1 includes canonical forms for EXTRA and NIDS as well as the versions with α and (α, μ) as additional tuning parameters.

Table 1: Different tunings of EXTRA [26] and NIDS [13] expressed as instances of Algorithm 1. Shown are default tunings, stepsize tuning (α), and overrelaxation (α, μ).

	α	β	γ	δ
EXTRA	$m(1-\sigma)/4L^2$	$1/2$	1	0
EXTRA(α)	α	$1/2$	1	0
EXTRA(α, μ)	α	$\mu/2$	μ	0
NIDS	$1/L$	$1/2$	1	$1/2$
NIDS(α)	α	$1/2$	1	$1/2$
NIDS(α, μ)	α	$\mu/2$	μ	$\mu/2$

3 Main Result

Our main theorem, Theorem 5, consists of a small convex semidefinite program (SDP) whose feasibility guarantees the linear convergence of a distributed algorithm in canonical form (1). The algorithm parameters $(\alpha, \beta, \gamma, \delta)$, problem data (κ, σ), and candidate linear rate ρ all appear as parameters in the SDP. Furthermore, the SDP has a fixed size that does not depend on n (the number of agents) or d (the dimension of the domain of f) and can thus be efficiently solved using a variety of established solvers.

Theorem 5 (Main analysis result)**.** *Consider the distributed optimization problem* (1) *solved using Algorithm 1. Suppose Assumptions 1 and 2 hold. Define the matrices*

$$M_0 := \begin{bmatrix} -2mL & L+m \\ L+m & -2 \end{bmatrix} \qquad \text{and} \qquad M_1 := \begin{bmatrix} \sigma^2 - 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

If there exist $P \in \mathbb{R}^{2\times 2}$ with $P = P^{\mathsf{T}} \succ 0$ and scalars $r \geq 0$ and ρ satisfying

$$\max\left(|1 - m\alpha|, |1 - L\alpha|\right) \leq \rho, \tag{4a}$$

$$\begin{bmatrix} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \end{bmatrix}^{\mathsf{T}} P \begin{bmatrix} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \end{bmatrix} - \rho^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^{\mathsf{T}} P \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 1 & 0 & 0 & -\delta \\ 0 & 0 & 1 & 0 \end{bmatrix}^{\mathsf{T}} M_0 \begin{bmatrix} 1 & 0 & 0 & -\delta \\ 0 & 0 & 1 & 0 \end{bmatrix} + r \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}} M_1 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \preceq 0, \tag{4b}$$

then there exists a constant $C > 0$ such that

$$\|x_i^k - x^\star\| \leq C\rho^k \tag{5}$$

for all agents $i \in \{1, \ldots, n\}$ and all iterations $k \geq 0$.

For fixed algorithm parameters $(\alpha, \beta, \gamma, \delta)$ and rate ρ, the SDP (4b) is a linear matrix inequality (LMI) in the variables (P, r) and therefore convex. Therefore, it is straightforward to check whether a given algorithm's convergence rate is upper-bounded by ρ. To find the best (smallest) upper bound, we observe that feasibility of (4) for some ρ_0 implies feasibility for all $\rho \geq \rho_0$. A bisection search on ρ is then guaranteed to find the minimal ρ, even though (4) is not jointly convex in (P, r, ρ). The core idea behind Theorem 3 is to posit a quadratic Lyapunov candidate of the form

$$V^k = \begin{bmatrix} x^k - (\mathbf{1}_n \otimes I_d)x^\star \\ w^k - w^\star \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} x^k - (\mathbf{1}_n \otimes I_d)x^\star \\ w^k - w^\star \end{bmatrix} \tag{6}$$

for some appropriate choice of $\widetilde{P}$. Feasibility of (4) can be shown to imply $V^{k+1} \leq \rho^2 V^k$, which ensures linear convergence of the distributed optimization algorithm when $\rho < 1$. A preliminary version of Theorem 5 appeared in [28] and involved two coupled SDPs of larger size than the one in Theorem 5.

Proof of Theorem 5. The first step involves combining (4a) and (4b) into a single semidefinite constraint. We can convert (4a) into the appropriate form by via the following lemma.

Lemma 6. *Suppose $0 < m \leq L$ and $\rho \geq 0$. Define M_0 as in Theorem 3. Then the following statements are equivalent:*

(i) $\max(|1 - m\alpha|, |1 - L\alpha|) \leq \rho$.

(ii) There exists $p_0 > 0$ such that: $\begin{bmatrix} 1 & -\alpha \end{bmatrix}^\mathsf{T} p_0 \begin{bmatrix} 1 & -\alpha \end{bmatrix} - \rho^2 \begin{bmatrix} 1 & 0 \end{bmatrix}^\mathsf{T} p_0 \begin{bmatrix} 1 & 0 \end{bmatrix} + M_0 \preceq 0$.

Proof. Consider the function $h(z) = |1 - \alpha z|$. Since $h(z)$ is a convex function of z, its maximum occurs at extreme values of z. Therefore, $\max(|1 - m\alpha|, |1 - L\alpha|) = \max_{m \leq z \leq L} h(z)$, and we may rewrite Item (i) equivalently as: $m \leq z \leq L \implies h(z)^2 \leq \rho^2$. Writing out each side explicitly as quadratic inequalities in z, we obtain: $(L - z)(z - m) \geq 0 \implies (1 - \alpha z)^2 \leq \rho^2$. By the lossless S-procedure (see for example [5, §B.2]), this implication is equivalent to Item (ii). ∎

Lemma 6 arises naturally in the derivation of the worst-case convergence rate of ordinary gradient descent [12, §4.4]. Indeed, (4a) can be interpreted as follows: distributed algorithms in the form of Algorithm 1 can converge no faster than centralized (non-distributed) gradient descent.

Applying Lemma 6 and multiplying the resulting constraint on the left and right by $\left[\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{smallmatrix}\right]^\mathsf{T}$ and $\left[\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{smallmatrix}\right]$, respectively, we have that (4a) is equivalent to the existence of some $p_0 > 0$ satisfying

$$\begin{bmatrix} 1 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T} \begin{bmatrix} p_0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} - \rho^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T} \begin{bmatrix} p_0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^\mathsf{T} M_0 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \preceq 0. \tag{7}$$

We next show that the semidefinite constraints (4b) and (7) can be combined using the following lemma, which appeared in [28, Lemma 6].

Lemma 7. *Suppose $X_1, X_2 \in \mathbb{R}^{n \times n}$. Suppose there exist $J_1, J_2 \in \mathbb{R}^{p \times p}$ that satisfy $J_1^2 = J_1$, $J_2^2 = J_2$, and $J_1 J_2 = J_2 J_1 = 0$. Then $X_1 \otimes J_1 + X_2 \otimes J_2 \preceq 0$ if and only if $X_1 \preceq 0$ and $X_2 \preceq 0$.*

One can check that $J_1 := \Pi \otimes I_d$ and $J_2 := (I - \Pi) \otimes I_d$ satisfy the requirements of Lemma 7. Therefore, (4b) and (7) hold if and only if there exists $P \succeq 0$, $r \geq 0$, and $p_0 > 0$ such that

$$\begin{aligned}
&\left(\begin{bmatrix} 1 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T} \begin{bmatrix} p_0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} - \rho^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T} \begin{bmatrix} p_0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right. \\
&\qquad \left. + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^\mathsf{T} M_0 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \right) \otimes J_1 \\
&\quad + \left(\begin{bmatrix} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \end{bmatrix}^\mathsf{T} P \begin{bmatrix} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \end{bmatrix} - \rho^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^\mathsf{T} P \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \right. \\
&\quad \left. + \begin{bmatrix} 1 & 0 & 0 & -\delta \\ 0 & 0 & 1 & 0 \end{bmatrix}^\mathsf{T} M_0 \begin{bmatrix} 1 & 0 & 0 & -\delta \\ 0 & 0 & 1 & 0 \end{bmatrix} + r \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\mathsf{T} M_1 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \otimes J_2 \preceq 0.
\end{aligned}$$

Distributing J_1 and J_2 using the mixed product property of the Kronecker product and combining like terms, we obtain the linear matrix inequality

$$\mathbf{V_1} - \rho^2 \mathbf{V_0} + \mathbf{M_0} + r\mathbf{M_1} \preceq 0 \tag{8}$$

where we have defined

$$\begin{aligned}
\widetilde{P} &:= \begin{bmatrix} p_0 & 0 \\ 0 & 0 \end{bmatrix} \otimes J_1 + P \otimes J_2, \\
\mathbf{V_1} &:= \begin{bmatrix} I_{nd} & \beta J_2 & -\alpha I_{nd} & -\gamma J_2 \\ \mathbf{0}_{nd} & J_2 & \mathbf{0}_{nd} & -J_2 \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} I_{nd} & \beta J_2 & -\alpha I_{nd} & -\gamma J_2 \\ \mathbf{0}_{nd} & J_2 & \mathbf{0}_{nd} & -J_2 \end{bmatrix}, \\
\mathbf{V_0} &:= \begin{bmatrix} I_{nd} & \mathbf{0}_{nd} & \mathbf{0}_{nd} & \mathbf{0}_{nd} \\ \mathbf{0}_{nd} & J_2 & \mathbf{0}_{nd} & \mathbf{0}_{nd} \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} I_{nd} & \mathbf{0}_{nd} & \mathbf{0}_{nd} & \mathbf{0}_{nd} \\ \mathbf{0}_{nd} & J_2 & \mathbf{0}_{nd} & \mathbf{0}_{nd} \end{bmatrix}, \\
\mathbf{M_0} &:= \begin{bmatrix} I_{nd} & \mathbf{0}_{nd} & \mathbf{0}_{nd} & -\delta J_2 \\ \mathbf{0}_{nd} & \mathbf{0}_{nd} & I_{nd} & \mathbf{0}_{nd} \end{bmatrix}^\mathsf{T} (M_0 \otimes I_{nd}) \begin{bmatrix} I_{nd} & \mathbf{0}_{nd} & \mathbf{0}_{nd} & -\delta J_2 \\ \mathbf{0}_{nd} & \mathbf{0}_{nd} & I_{nd} & \mathbf{0}_{nd} \end{bmatrix}, \\
\mathbf{M_1} &:= \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes J_2 \right)^\mathsf{T} (M_1 \otimes J_2) \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes J_2 \right).
\end{aligned}$$

Next, we define the following aggregated state variables to collect the variables stored and communicated by all agents:

$$x^k := \begin{bmatrix} x_1^k \\ x_2^k \\ \vdots \\ x_n^k \end{bmatrix}, \quad w^k := \begin{bmatrix} w_1^k \\ w_2^k \\ \vdots \\ w_n^k \end{bmatrix}, \quad u^k := \begin{bmatrix} \nabla f_1(x_1^k) \\ \nabla f_2(x_2^k) \\ \vdots \\ \nabla f_n(x_n^k) \end{bmatrix}, \quad v^k := \begin{bmatrix} v_1^k \\ v_2^k \\ \vdots \\ v_n^k \end{bmatrix}.$$

By Proposition 3, there exists a fixed point $(x_i^\star, w_i^\star, u_i^\star, v_i^\star, y_i^\star)$ of Algorithm 1 with $x_i^\star = x^\star$ for all $i \in \{1, \ldots, n\}$. Since the local objective functions are strongly convex from Assumption 1, this fixed point is unique and $x^\star$ is the optimizer of (1). Denoting the aggregated fixed point as $(\mathbf{1}x^\star, w^\star, u^\star, v^\star, y^\star)$ where $\mathbf{1} := 1_n \otimes I_d$, we can multiply (8) on the left and right by

$\begin{bmatrix}(x-\mathbf{1}x^\star)^\mathsf{T} & (w-w^\star)^\mathsf{T} & (u-u^\star)^\mathsf{T} & (v-v^\star)^\mathsf{T}\end{bmatrix}^\mathsf{T}$ and its transpose, respectively, to obtain

$$\begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \\ u^k-u^\star \\ v^k-v^\star \end{bmatrix}^\mathsf{T} (\mathbf{V_1}-\rho^2\mathbf{V_0}+\mathbf{M_0}+\mathbf{M_1}) \begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \\ u^k-u^\star \\ v^k-v^\star \end{bmatrix} \leq 0.$$

Due to the initialization of the algorithm, $(1_n \otimes 1_d)^\mathsf{T} w^k = 0$ and consequently $J_2 w^k = w^k$. Also, recognize that $J_2 v^k = v^k$ since $1_n^\mathsf{T}\mathcal{L}^k = 0_n^\mathsf{T}$ from Assumption 2. This leads to the simplification

$$\begin{bmatrix} x^{k+1}-\mathbf{1}x^\star \\ w^{k+1}-w^\star \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} x^{k+1}-\mathbf{1}x^\star \\ w^{k+1}-w^\star \end{bmatrix} - \rho^2 \begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \end{bmatrix}$$
$$+ \begin{bmatrix} y^k-\mathbf{1}y^\star \\ u^k-u^\star \end{bmatrix}^\mathsf{T} (M_0 \otimes I_{nd}) \begin{bmatrix} y^k-\mathbf{1}y^\star \\ u^k-u^\star \end{bmatrix} + r \begin{bmatrix} x^k-\mathbf{1}x^\star \\ v^k-v^\star \end{bmatrix}^\mathsf{T} (M_1 \otimes J_2) \begin{bmatrix} x^k-\mathbf{1}x^\star \\ v^k-v^\star \end{bmatrix} \leq 0. \quad (9)$$

By Propositions 1 and 2, the third and fourth terms are nonnegative. It follows that

$$\begin{bmatrix} x^{k+1}-\mathbf{1}x^\star \\ w^{k+1}-w^\star \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} x^{k+1}-\mathbf{1}x^\star \\ w^{k+1}-w^\star \end{bmatrix} \leq \rho^2 \begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \end{bmatrix}.$$

Iterating, we obtain

$$\begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \end{bmatrix} \leq \rho^{2k} \begin{bmatrix} x^0-\mathbf{1}x^\star \\ w^0-w^\star \end{bmatrix}^\mathsf{T} \widetilde{P} \begin{bmatrix} x^0-\mathbf{1}x^\star \\ w^0-w^\star \end{bmatrix}.$$

Letting $\mathrm{cond}(\widetilde{P}) := \lambda_{\max}(\widetilde{P})/\lambda_{\min}(\widetilde{P})$, the ratio of largest to smallest eigenvalue of $\widetilde{P}$, we can deduce the following bound for all agents $i \in \{1,\dots,n\}$ and all iterations $k \geq 0$.

$$\left\| \begin{bmatrix} x^k-\mathbf{1}x^\star \\ w^k-w^\star \end{bmatrix} \right\|^2 \leq \rho^{2k}\,\mathrm{cond}(\widetilde{P}) \left\| \begin{bmatrix} x^0-\mathbf{1}x^\star \\ w^0-w^\star \end{bmatrix} \right\|^2$$

Thus, $\|x_i^k - x^\star\| \leq C\rho^k$ for some constant $C > 0$ independent of k. ∎

4 Algorithm Design

We now use the SDP (4) to design our algorithm, which we denote as SVL. Our algorithm has the form of Algorithm 1 where the parameters $(\alpha, \beta, \gamma, \delta)$ are chosen to optimize the worst-case convergence rate ρ subject to the SDP being feasible. To motivate the structure of our algorithm, we show how it corresponds to an inexact version of the alternating direction method of multipliers (ADMM), as well as how it reduces to well-known consensus and optimization algorithms in special cases.

The problem of minimizing the worst-case convergence rate ρ over the algorithm parameters $(\alpha, \beta, \gamma, \delta)$ and SDP solution (P, r) subject to the SDP being feasible is difficult due to the nonlinear matrix inequality (4b). Instead, we show that for a particular choice of (α, γ, δ), the remaining parameters (β, ρ) can be chosen such that the matrix in (4b) is rank one. To justify this choice of parameters, we show in Section 4.2 that this corresponds to an inexact version of ADMM. Furthermore, we have performed extensive numerical optimizations of the SDP which suggest that the optimal parameters do in fact have this structure. We now state our main design result which describes the SVL algorithm.

Theorem 8 (SVL)**.** *Consider applying Algorithm 1 to the distributed optimization problem* (1)*, and suppose Assumptions 1 and 2 hold. Define* $\eta := 1 + \rho - \kappa\,(1-\rho)$ *and choose the parameters*

$$\alpha = \frac{1-\rho}{m}, \qquad \gamma = 1+\beta, \qquad \delta = 1, \tag{10}$$

where β *and* ρ *satisfy the constraints*

$$\big(2\beta - (1-\rho)(\kappa+1)\big)(\beta - 1 + \rho^2) < 0, \tag{11a}$$

$$\rho^2 \left(\frac{\beta - 1 + \rho^2}{\beta - 1 + \rho}\right)\left(\frac{2 - \eta - 2\beta}{2\rho^2\beta - (1-\rho^2)\eta}\right)\left(\frac{(2\rho^2+\eta)\beta - (1-\rho^2)\eta}{(1+\rho)(\eta - 2\eta\rho + 2\rho^2) - (2\rho^2+\eta)\beta}\right) = \sigma^2. \tag{11b}$$

Then the bound (5) *holds for all agents* $i \in \{1, \dots, n\}$ *and all iterations* $k \geq 0$*.*

4.1 Optimizing the algorithm parameters

Theorem 8 provides conditions on the parameters $(\alpha, \beta, \gamma, \delta)$ of Algorithm (1) such that the algorithm converges with rate at least ρ. The theorem, however, does not address the problem of optimizing the convergence rate since β and ρ must only be chosen to satisfy the constraints (11). This is because the optimal parameters do not admit a closed-form solution for the convergence rate ρ as a function of the spectral gap σ and function parameters m and L. However, we now provide a systematic method for computing the optimal parameters.

The parameters must satisfy (11b), but this equation does not have a closed-form solution for ρ. Instead, we consider fixing the spectral radius and maximizing the corresponding spectral gap. We can then choose the parameter β to maximize the spectral gap σ^2 in (11b). Setting the derivative equal to zero, we find that the value of β which maximizes σ^2 for a fixed convergence rate ρ satisfies

$$0 = \frac{\mathrm{d}\sigma^2}{\mathrm{d}\beta} \quad \Longrightarrow \quad 0 = \big(\beta\,\big(1 - \kappa + 2\rho\,(1+\rho)\big) - \eta\,(1-\rho^2)\big)\big(s_0 + s_1\,\beta + s_2\,\beta^2 + s_3\,\beta^3\big)$$

where the coefficients s_i are given by

$$\begin{aligned}
s_0 &:= \eta\,\big(1-\rho^2\big)^2\big(\eta - (3-\eta)\eta\rho + 2\,(1-\eta)\rho^2 + 2\rho^3\big),\\
s_1 &:= -\big(1-\rho^2\big)\Big(\eta^3\rho + 4\rho^5 - 2\eta\rho^2\,(2\rho^2 + \rho - 3) + \eta^2\,\big(4\rho^3 - 4\rho^2 - 6\rho + 3\big)\Big),\\
s_2 &:= 3\eta\,(1-\rho)^2(1+\rho)(2\rho^2+\eta),\\
s_3 &:= (2\rho^2+\eta)(2\rho^3 - \eta),
\end{aligned}$$

where $\eta := 1 + \rho - \kappa\,(1-\rho)$. Solving the first factor for β, we find that it does not satisfy the inequality (11a) and is therefore not a valid solution. The optimal β must then make the second factor zero. Therefore, we can do a bisection search over ρ where at each iteration of the bisection search we solve the cubic equation

$$s_0 + s_1\,\beta + s_2\,\beta^2 + s_3\,\beta^3 = 0 \tag{12}$$

to find the unique real solution β which satisfies (11a). Substituting this value for β into (11b) we can solve for σ. If this value is less than σ, we increase ρ; otherwise, we decrease ρ. We then repeat this procedure until σ is sufficiently close to the spectral gap. We summarize this procedure for

Algorithm 2 (computing the SVL parameters)

Initialization: Let $0 < m \leq L$, $0 \leq \sigma < 1$, and $\varepsilon > 0$. Define $\kappa := L/m$. Set $\rho_1 = 0$ and $\rho_2 = 1$.
while $\rho_2 - \rho_1 > \varepsilon$ **do**
  $\rho = (\rho_1 + \rho_2)/2$
  Let β be the unique real solution to the cubic (12) that satisfies (11a).
  Using this value of β, let $\hat{\sigma}$ denote the solution to (11b).
  if $\hat{\sigma} < \sigma$ **then**
    $\rho_1 = \rho$
  else
    $\rho_2 = \rho$
  end if
end while

finding the parameters β and ρ which optimize the worst-case convergence rate in Algorithm 2; we refer to Algorithm 1 using these optimal parameters along with those in (10) as SVL.

Using this procedure for computing the worst-case convergence rate of SVL, Figure 2 displays ρ as a function of the spectral gap σ and the centralized gradient rate $\frac{\kappa-1}{\kappa+1}$. One of the remarkable aspects of the SVL algorithm is that it actually achieves the same worst-case convergence rate as *centralized* gradient descent if the spectral gap is sufficiently small. In this case, there is sufficient mixing among the agents so that the convergence rate is limited by the difficulty of the optimization problem and not the problem of having agents agree on the solution (i.e., consensus). This corresponds to scenario **(a)** in Theorem 8 and results in the horizontal lines for small values of σ in the left panel of Figure 2. Viewed another way, the convergence rate is limited by the difficulty of the optimization problem when the problem is ill-conditioned (i.e., κ is large) which corresponds to the curves approaching the straight line at $\rho = \frac{\kappa-1}{\kappa+1}$ in the right panel of Figure 2.

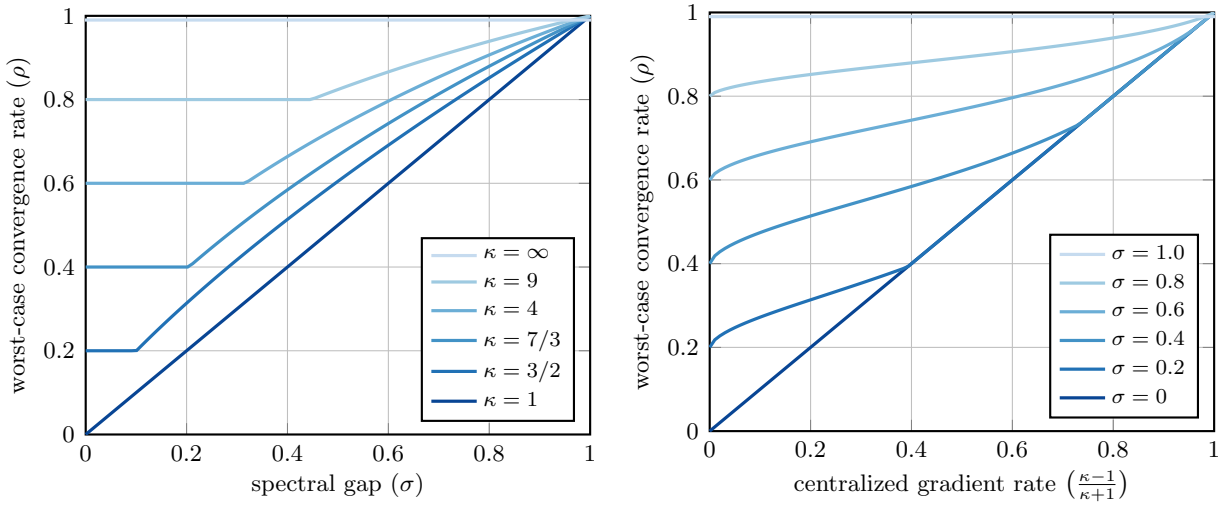

Figure 2: Worst-case convergence rate of SVL in Theorem 8.

4.2 Interpretation of SVL as inexact ADMM

To motivate the structure of the SVL algorithm, we now show how it can be interpreted as an inexact version of the alternating direction method of multipliers (ADMM). Using the formulation in [4, Section 7.1], the optimization problem (1) can be solved using the ADMM algorithm

$$x_i^{k+1} = \arg\min_{x_i} f(x_i) + (z_i^k)^\mathsf{T}(x_i - y_i^k) + \tfrac{\beta}{2}\|x_i - y_i^k\|^2 \tag{13a}$$

$$y_i^{k+1} = \frac{1}{n}\sum_{j=1}^{n} x_j^{k+1} \tag{13b}$$

$$z_i^{k+1} = z_i^k + \beta\,(x_i^{k+1} - y_i^{k+1}) \tag{13c}$$

where (x_i^k, y_i^k, z_i^k) are the variables associated with agent i at time k, and β is the ADMM parameter. To implement this algorithm, however, each agent must solve the local optimization problem (13a) *exactly* as well as compute the *exact* average (13b) at each iteration. Instead, we consider a variant where the computations and communications are *inexact*. Specifically, we replace the exact minimization (13a) with a single gradient step with initial condition y_i^k and stepsize $\alpha > 0$, and we replace the exact averaging step (13b) with a single gossip step using the Laplacian matrix $\mathcal{L}^k$. This gives the following inexact version of ADMM:

$$\begin{aligned} x_i^{k+1} &= y_i^k - \alpha\left(\nabla f(y_i^k) + z_i^k\right) \\ y_i^{k+1} &= x_i^{k+1} - \sum_{j=1}^{n} \mathcal{L}_{ij}^{k+1}\, x_j^{k+1} \\ z_i^{k+1} &= z_i^k + \beta\,(x_i^{k+1} - y_i^{k+1}) \end{aligned}$$

Defining the state $w_i^k := -\frac{\alpha}{\beta} z_i^{k-1}$, this algorithm is equivalent to Algorithm 1 with parameters $\gamma = 1 + \beta$ and $\delta = 1$. In other words, the SVL algorithm corresponds to an inexact version of ADMM where α is the stepsize of the gradient step and β is the ADMM parameter.

4.3 Special cases

We now show how the SVL algorithm reduces to well-known consensus and optimization algorithms in special cases.

Case: $n = 1$. When there is only a single agent, the distributed optimization problem (1) is equivalent to a centralized optimization problem. In this case, the Laplacian matrix is simply the scalar $\mathcal{L}^k \equiv 0$, so $v_1^k = 0$ for all $k \geq 0$. We then have $w_1^k = 0$ for all $k \geq 0$ since $w_1^{k+1} = w_1^k - v_1^k$ with initial condition $w_1^0 = 0$. Algorithm 1 then simplifies to

$$x_1^{k+1} = x_1^k - \alpha\,\nabla f(x_1^k), \qquad x_1^0 \text{ arbitrary},$$

which is ordinary gradient descent with stepsize α. The fastest possible gradient rate is $\rho = \frac{\kappa-1}{\kappa+1}$ and is achieved when $\alpha = \frac{2}{L+m}$.

Case: $\kappa = 1$. When the condition ratio is unity (i.e., $m = L$), the distributed optimization problem (1) is equivalent to average consensus. In this case, the parameters of SVL are simply $\alpha = \frac{1}{L}$, $\beta = 1$, $\gamma = 2$, and $\delta = 1$. Also, the objective functions are quadratic, so we may assume without loss of generality that they have the form $f_i(x) = \frac{L}{2}\|x - r_i\|^2$ where $r_i \in \mathbb{R}^d$ is a parameter on agent $i \in \{1, \dots, n\}$. The SVL algorithm then simplifies to

$$x_i^{k+1} = x_i^k - \sum_{j=1}^{n} \mathcal{L}_{ij}^k \, x_j^k, \qquad x_i^0 = r_i,$$

which is average consensus on the parameters $\{r_i\}_{i=1}^n$; see [30,31]. This is often referred to as *static* average consensus since the reference signals only affect the initial condition of the algorithm. Alternatively, we can obtain a *dynamic* average consensus algorithm if we allow the objective functions to vary in time; see [11] for an overview of dynamic average consensus. In particular, suppose the objective functions have the form $f_i^k(x) = \frac{L}{2}\|x - r_i^k\|^2$ where $r_i^k \in \mathbb{R}^d$ is the (time-varying) reference signal on agent $i \in \{1, \dots, n\}$ at time k. In this case, SVL simplifies to

$$x_i^{k+1} = x_i^k - \sum_{j=1}^{n} \mathcal{L}_{ij}^k \, x_j^k + \left(r_i^k - r_i^{k-1}\right), \qquad x_i^0 = r_i^0,$$

which is a dynamic average consensus algorithm since the reference signals are continually injected into the dynamics. In general, the signals converge to a neighborhood of the time-varying average whose size depends on how quickly the reference signals vary in time [40]; the exact average is recovered for constant signals since the algorithm reduces to static consensus in that case. Moreover, the worst-case rate of convergence for consensus rate of constant signals is $\rho = \sigma$ [31].

5 Numerical Results

In this section, we compare the worst-case performance of SVL with other first-order distributed algorithms. Since our analysis method is a guarantee of an upper bound on the convergence rate, we show empirically matching lower bounds that suggest our certified convergence rates are tight. To construct the worst-case trajectory we optimize the communication graph at each step of the algorithm. Figure 3 plots algorithm trajectories via this adversarial optimization method and shows that the examples we generate achieve the worst-case bounds certified by SDP (4) for NIDS, EXTRA, and SVL.

5.1 Algorithm comparison

We compared the worst-case performance of SVL and the six algorithms in Table 1, and the results are shown in Figure 1. For each algorithm, we used a bisection search to find the smallest rate ρ that yielded a feasible solution to the SDP (4). We implemented the SDP in Julia [3] with the JuMP [6] modeling package and the Mosek solver [1]. In the case of (α) and (α, μ) variants, we performed a parameter search and used the parameters that yielded the smallest possible ρ. Specifically, we used Brent's method and the Nelder–Mead method, respectively, as implemented in the Optim package [14] with warm-start as σ ranged from 0 to 1.

As shown in Figure 1, different tunings of EXTRA and NIDS produce different worst-case depending on the values value of σ. The default tunings can be substantially improved if optimized

stepsizes α or overrelaxation parameters μ are used. Our proposed SVL algorithm outperforms the other methods we tested. This comes as no surprise since it was designed by optimizing over instances of Algorithm 1, which subsumes EXTRA and NIDS.

Also shown in Figure 1 are the lower bounds described by the special cases in Section 4.3. Specifically, the case $n = 1$ of a single agent reduces to ordinary gradient descent from which $\rho \geq \frac{\kappa-1}{\kappa+1}$. Also, the case $\kappa = 1$ reduces to average consensus from which $\rho \geq \sigma$. SVL matches the lower bound for small σ, but there is a gap when σ gets larger. NIDS also matches the lower bound for small σ, but only when the stepsize is optimized.

5.2 Generating worst-case examples

Given problem data (κ, σ), we would like to simulate algorithm trajectories that achieve the worst-case rate certified by a solution to (4). We devise a procedure to construct such examples for a desired rate ρ using the solution (P, r) from (4). First, we fix the local function of each agent $i \in \mathcal{V}$ to be a positive definite quadratic of the form $f_i(x_i) = x_i^\mathsf{T} Q_i x_i$ where $mI_d \preceq Q_i \preceq LI_d$ and Q_i has condition ratio κ. At iteration k of the algorithm, we choose the vector v^k to be the solution of the optimization problem

$$\begin{array}{ll} \max\limits_{v \in \mathbb{R}^{nd}} & V^{k+1} - \rho^2 V^k \\ \text{s.t.} & \begin{bmatrix} x^k - (1_n \otimes I_d)x^\star \\ v^k \end{bmatrix}^\mathsf{T} \left(M_1 \otimes (I_n - \Pi) \otimes I_d\right) \begin{bmatrix} x^k - (1_n \otimes I_d)x^\star \\ v^k \end{bmatrix} \geq 0 \end{array} \tag{14}$$

where V^k is defined in (6). The objective functions satisfy Assumption 1 by construction, and the constraint in (14) ensures that the spectral gap of the Laplacian matrix at each step is upper-bounded by σ so that Assumption 2 holds. Therefore, the trajectory produced by (14) satisfies both our assumptions. Furthermore, the third term in (9), corresponding to the strong convexity and smoothness of the local functions, is identically zero because the f_i are each set to have condition ratio κ. To obtain the worst-case trajectory, we maximize the difference in the Lyapunov function given by the cost $V^{k+1} - \rho^2 V^k$. If the SDP (4) is feasible, then $V^{k+1} \leq \rho^2 V^k$, so the objective is nonpositive. If we can obtain $V^{k+1} = \rho^2 V^k$ at each iteration, then the trajectory converges linearly with exactly rate ρ.

Problem (14) is a quadratically constrained quadratic program (QCQP) but is nonconvex since the objective function is a difference of convex functions. At each time step k, the aggregated state variables x^k and w^k are known from the previous iteration. Furthermore, the aggregated gradient vector of all agents is $u^k = Qy^k = Q(x^k - \delta v^k)$, where $Q = \operatorname{diag}(Q_1, \ldots, Q_n)$. Then both the cost function and constraint in (14) are quadratic functions in v^k. Written in explicit form, (14) becomes

$$\begin{array}{ll} \max\limits_{v} & v^\mathsf{T} A_0 v + 2b_0^\mathsf{T} v + c_0 \\ \text{s.t.} & v^\mathsf{T} A_1 v + 2b_1^\mathsf{T} v + c_1 \geq 0. \end{array} \tag{15}$$

This single constraint optimization problem is discussed in [5]. The authors relax (15) to obtain

$$\begin{array}{ll} \max\limits_{X} & \operatorname{tr}(A_0 X) + 2b_0^\mathsf{T} v + c_0 \\ \text{s.t.} & \operatorname{tr}(A_1 X) + 2b_1^\mathsf{T} v + c_1 \geq 0 \\ & \begin{bmatrix} X & v \\ v^\mathsf{T} & 1 \end{bmatrix} \succeq 0 \end{array} \tag{16}$$

where $X \in \mathbb{R}^{nd \times nd}$ is the optimization variable. If $X = vv^\mathsf{T}$, then (16) is equivalent to (15) since $v^\mathsf{T} A_j v = \mathrm{tr}(A_j X)$. Strong duality holds between (15) and (16) if (15) is strictly feasible [5], so we may solve (16) at each step to find v^k.

Remark 9. *The problem* (16) *is often ill-conditioned and hence we rescale the cost function during implementation so that* $c_0 = 1$. *We also remove the directions corresponding to the zero eigenvalue from the constraint by taking a truncated singular value decomposition of* A_1. *This prevents the solver from yielding unbounded solutions.*

This procedure of solving for v^k at each step generates examples in simulation that achieve the worst-case bounds from (4). We simulate EXTRA, NIDS, and SVL for a condition ratio of κ and several values of σ in Figure 3. Each trajectory plots the total error, $\|x^k - (1_n \otimes I_d)x^\star\|$, and has a slope that aligns with the rate bound from (4).

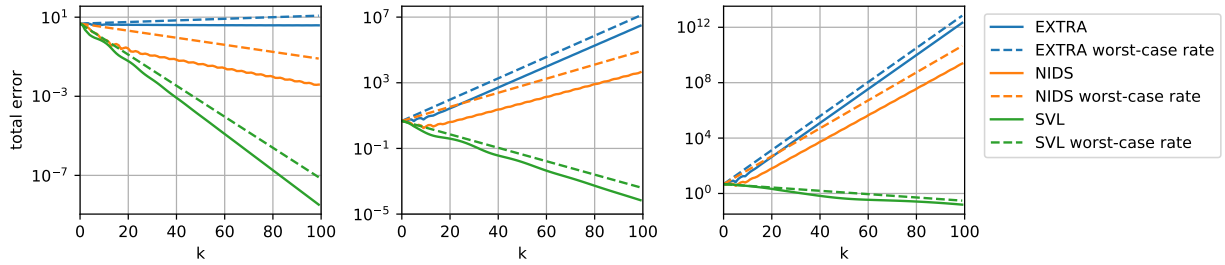

Figure 3: Worst-case trajectories for EXTRA, NIDS, and SVL. Simulations are performed for $\kappa = 10$, $n = 10$, $d = 2$, and $\sigma \in \{0.5, 0.7, 0.9\}$ (from left to right).

Remark 10. *Note that our procedure to generate worst-case examples finds* v^k *given* x^k *and not* $\mathcal{L}^k$ *explicitly. If we transform coordinates from* v^k *to the a vector involving the entries of* $\mathcal{L}^k$, *problem* (16) *is still feasible. However, the resulting* $\mathcal{L}^k$ *does not necessarily satisfy the spectral gap bound* σ. *This is because problem* (16) *certifies that* $\|(I - \Pi - \mathcal{L}^k)x^k\| \leq \sigma\|x^k\|$, *but the bound may not hold for arbitrary* $x \in^{nd}$. *In other words, the gain of* $I - \Pi - \mathcal{L}^k$ *is bounded by* σ *in the direction of the current iterate* x^k, *but it may not be bounded by* σ *in all directions. To circumvent this issue, instead of the linear Laplacian matrix we could use the nonlinear Laplacian operator*

$$\mathcal{L}^k(x) = \begin{cases} v^k, & x = x^k \\ 0, & \textit{otherwise} \end{cases},$$

which does satisfy the norm bound since it is zero in all directions except for x^k.

6 Conclusion

We presented a universal analysis framework for a broad class of first-order distributed optimization algorithms over time-varying graphs. The framework provides worst-case certificates of linear convergence via semidefinite programming, and we show empirically that our rate bounds are likely tight. Optimizing the SDP from our analysis framework, we designed a novel distributed algorithm, SVL, which outperforms EXTRA and NIDS in the worst case, regardless of how they are tuned.

References

[1] APS Mosek. The MOSEK optimization software, 2010. Online at `http://www.mosek.com`.

[2] J. A. Bazerque and G. B. Giannakis. Distributed spectrum sensing for cognitive radio networks by exploiting sparsity. *IEEE Transactions on Signal Processing*, 58(3):1847–1862, 2009.

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

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, volume 3. Foundations and Trends in Machine Learning, 2010.

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

[6] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[7] M. Fazlyab, S. Paternain, A. Ribeiro, and V. M. Preciado. Distributed smooth and strongly convex optimization with inexact dual methods. In *American Control Conference*, pages 3768–3773, 2018.

[8] P. A. Forero, A. Cano, and G. B. Giannakis. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11:1663–1707, 2010.

[9] D. Jakovetić. A unification, generalization, and acceleration of exact distributed first order methods. *arXiv preprint arXiv:1709.01317*, 2017.

[10] B. Johansson. *On distributed optimization in networked systems*. PhD thesis, KTH, 2008.

[11] S. S. Kia, B. Van Scoy, J. Cortés, R. A. Freeman, K. M. Lynch, and S. Martínez. Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. volume 39, pages 40–72, 2019.

[12] 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.

[13] Z. Li, W. Shi, and M. Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *arXiv preprint arXiv:1704.07807*, 2017.

[14] P. K. Mogensen and A. N. Riseth. Optim: A mathematical optimization package for julia. *Journal of Open Source Software*, 3(24), 2018.

[15] A. Nedić. Asynchronous broadcast-based convex optimization over a network,. *IEEE Transactions on Automatic Control*, 56(6):1337–1351, 2011.

[16] A. Nedić, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[17] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[18] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004.

[19] J. Ortega and W. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic press, 1970.

[20] G. Qu and N. Li. Accelerated distributed Nesterov gradient descent for smooth and strongly convex functions. In *Allerton Conference on Communication, Control, and Computing*, pages 209–216, 2016.

[21] G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 2017.

[22] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 20–27. ACM, 2004.

[23] S. S. Ram, V. V. Veeravalli, and A. Nedić. Distributed non-autonomous power control through distributed convex optimization. In *IEEE INFOCOM*, pages 3001–3005, 2009.

[24] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3027–3036, 2017.

[25] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2740–2749, 2018.

[26] W. Shi, Q. Ling, G. Wu, and W. Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[27] Y. Sun, A. Daneshmand, and G. Scutari. Convergence rate of distributed optimization algorithms based on gradient tracking. *arXiv:1905.02637*, 2019.

[28] A. Sundararajan, B. Hu, and L. Lessard. Robust convergence analysis of distributed optimization algorithms. In *Allerton Conference on Communication, Control, and Computing*, pages 1206–1212, 2017.

[29] A. Sundararajan, B. Van Scoy, and L. Lessard. A canonical form for first-order distributed optimization algorithms. In *American Control Conference*, pages 4075–4080, 2019.

[30] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, Massachusetts Institute of Technology, 1984.

[31] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[32] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing*, 67(1):33–46, 2007.

[33] R. Xin, D. Jakovetić, and U. A. Khan. Distributed nesterov gradient methods over arbitrary graphs. *IEEE Signal Processing Letters*, 2019.

[34] R. Xin and U. A. Khan. Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking. *arXiv:1808.02942*, 2018.

[35] R. Xin and U. A. Khan. A linear algorithm for optimization over directed graphs with geometric convergence. *IEEE Control Systems Letters*, 2(3):315–320, 2018.

[36] J. Xu, S. Zhu, Y. C. Soh, and L. Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *IEEE Conference on Decision and Control*, pages 2055–2060, 2015.

[37] J. Xu, S. Zhu, Y. C. Soh, and L. Xie. Convergence of asynchronous distributed gradient methods over stochastic networks. *IEEE Transactions on Automatic Control*, 63(2):434–448, 2018.

[38] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed. Exact diffusion for distributed optimization and learning—Part I: Algorithm development. *arXiv preprint arXiv:1702.05122*, 2017.

[39] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed. Exact diffusion for distributed optimization and learning—Part II: Convergence analysis. *arXiv preprint arXiv:1702.05142*, 2017.

[40] M. Zhu and S. G. Martinez. Discrete-time dynamic average consensus. *Automatica*, 46:322–329, 2010.

A Appendix

A.1 Proof of Proposition 2

Since $\mathcal{L}^k 1_n = 0_n$, we have: $I - \Pi - \mathcal{L}^k = (I - \Pi - \mathcal{L}^k)(I - \Pi)$. Then by the definition of the matrix norm and Assumption 2,

$$\sigma \geq \left\|I - \Pi - \mathcal{L}^k\right\| = \left\|(I - \Pi - \mathcal{L}^k)(I - \Pi)\right\| = \max_{y \in \mathbb{R}^n, y \neq 0} \frac{\left\|(I - \Pi - \mathcal{L}^k)(I - \Pi)y\right\|}{\|y\|}.$$

Without loss of generality, we may write $y = \Pi z + (I - \Pi)w$ where z and w are arbitrary. By orthogonality, we have $\|y\|^2 = \|\Pi z\|^2 + \|(I - \Pi)w\|^2$. Substituting the decomposition of y into the above equation, we obtain

$$\begin{aligned}\sigma &\geq \max_{w,z \in \mathbb{R}^n, y \neq 0} \frac{\left\|(I - \Pi - \mathcal{L}^k)(I - \Pi)w\right\|}{\sqrt{\|\Pi z\|^2 + \|(I - \Pi)w\|^2}} = \max_{w \in \mathbb{R}^n, y \neq 0} \frac{\left\|(I - \Pi - \mathcal{L}^k)(I - \Pi)w\right\|}{\|(I - \Pi)w\|} \\ &= \max_{w \in \mathbb{R}^n, y \neq 0} \frac{\left\|(I - \Pi)(w - \mathcal{L}^k w)\right\|}{\|(I - \Pi)w\|}\end{aligned}$$

where the last two steps follow because the maximum is attained with $z = 0$ and because $\mathcal{L}^k 1_n = 0_n$ and hence $\mathcal{L}^k \Pi = \Pi \mathcal{L}^k = \mathbf{0}$. Squaring both sides and rewriting as a quadratic form yields

$$\begin{bmatrix} w \\ \mathcal{L}^k w \end{bmatrix}^\mathsf{T} \left(\begin{bmatrix} \sigma^2 - 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes (I - \Pi) \right) \begin{bmatrix} w \\ \mathcal{L}^k w \end{bmatrix} \geq 0. \tag{17}$$

Setting $u := \begin{bmatrix} w_1^\mathsf{T} & \dots & w_d^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and summing d instances of (17) with $w \in \{w_1, \dots, w_d\}$ yields

$$\begin{bmatrix} u \\ (\mathcal{L}^k \otimes I_d)u \end{bmatrix}^\mathsf{T} \left(\begin{bmatrix} \sigma^2 - 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes (I - \Pi) \otimes I_d \right) \begin{bmatrix} u \\ (\mathcal{L}^k \otimes I_d)u \end{bmatrix} \geq 0.$$

Setting $u \mapsto x^k - (1_n \otimes I_d)x^\star$ completes the proof. ∎

A.2 Proof of Proposition 3

The first statement of Proposition 3 is proved in Theorem 4 in [29]. We now prove the second statement. By Assumption 2, to achieve stationarity, $v_i^\star = 0$, $y_i^\star = x_i^\star$, $x_i = x_j$ since the graph is connected. Summing (C.4) gives that $\alpha \sum_{i=1}^n u_i^\star = \beta \sum_{i=1}^n w_i^\star$. By (C.5), the initialization of the algorithm, and $\alpha \neq 0$, it follows that $\sum_{i=1}^n u_i^\star = 0$, so the fixed point solves (1). With $x^\star$ as the solution to (1), we can construct a fixed point as follows: $x_i^\star = x^\star$, $w_i^\star = (\alpha/\beta)u_i^\star$, where we have used $\beta \neq 0$, $u_i^\star = \nabla f_i(x^\star)$, $v_i^\star = 0$. ∎

A.3 Proof of Theorem 8

Suppose we choose the parameters in (10) where β and ρ satisfy (11). Then the inequality (4a) is satisfied since $\alpha = \frac{1-\rho}{m}$ with $\rho \in \left[\frac{\kappa-1}{\kappa+1}, 1\right)$. Now consider the potential SDP solution

$$P = \frac{t_3}{\alpha^2 \rho^2} \begin{bmatrix} 1 + \rho^2 \dfrac{t_1}{t_4} & -1 \\ -1 & 1 \end{bmatrix} \qquad \text{and} \qquad r = \frac{t_5}{\alpha^2 t_2}$$

where

$$
\begin{aligned}
t_1 &= 2\,(1-\beta) - \eta, \\
t_2 &= \beta - 1 + \rho^2, \\
t_3 &= \beta\,(\eta + 2\rho^2) - \eta\,(1-\rho^2), \\
t_4 &= 2\beta\rho^2 - \eta\,(1-\rho^2), \\
t_5 &= (1-\beta-\rho)\big(\beta\,(\eta+2\rho^2) - (1-\rho^2)(1-\kappa+2\kappa\rho)\big).
\end{aligned}
$$

Using these values along with the value for σ^2 in (11b), the matrix in (4b) is equal to the rank-one matrix $-\frac{1}{t_2 t_4} z z^{\mathsf{T}}$ where

$$
z = \frac{1}{\alpha\rho}\begin{bmatrix} \big(2-\alpha\,(L+m)\big)(1-\rho^2)^2 - \big(2\,(1-\rho^4) - \alpha\,(L+m)\big)\beta \\ -t_2 t_3 \\ \alpha t_2\,\big(2-\alpha\,(L+m)\big) \\ \beta\,\big(t_3 - \alpha\rho^2(L+m)\big) \end{bmatrix}.
$$

In order for this to be a valid solution, we must have $t_3 > 0$ and $t_1/t_4 > 0$ (so that $P \succ 0$), $t_5/t_2 \geq 0$ (so that $r \geq 0$), and $t_2 t_4 > 0$ (so that (4b) holds). All of these inequalities hold if and only if (11a) holds. Therefore, the SDP has a rank-one solution using the parameters in (10) if β and ρ satisfy (11). The convergence bound (5) then holds from Thm. 5. ∎