ANALYSIS AND DESIGN OF DISTRIBUTED OPTIMIZATION ALGORITHMS

by

Akhil Sundararajan

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2021

Date of final oral examination: 04/22/2021

The dissertation is approved by the following members of the Final Oral Committee:
Laurent Lessard, Associate Professor, Electrical and Computer Engineering
William A. Sethares, Professor, Electrical and Computer Engineering
Dominic Gross, Assistant Professor, Electrical and Computer Engineering
Xiangru Xu, Assistant Professor, Mechanical Engineering

To Appa, Amma, and Arvind

ACKNOWLEDGMENTS

I am grateful to my advisor, Prof. Laurent Lessard, for his constant inspiration and support. Laurent is a profound researcher, thoughtful teacher, and compassionate mentor. I feel privileged and humbled to have had the opportunity to pursue my doctoral research under his guidance.

I thank Prof. William Sethares, Prof. Dominic Gross, and Prof. Xiangru Xu for being on my doctoral committee and providing me with useful comments and feedback.

Thank you to Bryan Van Scoy and Bin Hu for their mentorship and technical discussions. I would also like to thank Saman Cyrus, Mruganka Kashyap, and Emad Sadeghi who were an integral part of my research experience through lab meetings amd informal discussions alike.

Gratitude to the National Science Foundation for funding support for my research.

It is because of my family's unconditional love and encouragement that I could make it through graduate school. I would like to thank my parents, Sundar and Bharati, and my brother, Arvind. I am truly blessed to have a family who supports me and always helps me to perform at my best.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

I_n	identity matrix in $\mathbb{R}^{n\times n}$
e_i	ith column of identity matrix
1_n	column vector of all ones in $\mathbb{R}^n$
Π	$\frac{1}{n}1_n 1_n^\mathsf{T}$, projection matrix onto 1_n
$0_{p\times q}$	$p \times q$ zeros matrix
$\otimes$	Kronecker matrix product
$\|x\|$	standard Euclidean norm of a vector x
$\|A\|$	spectral norm of a matrix A
$\|x\|_P$	P-norm of a vector, $(x^\mathsf{T}Px)^{1/2}$
κ	condition ratio of a function
σ	spectral gap of a graph
$\mathcal{G}(\mathcal{V},\mathcal{E})$	graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$
$\mathcal{F}(m,L)$	class of sector-bounded functions parameterized by (m,L)

ABSTRACT

This work concerns the analysis and design of distributed first-order optimization algorithms. The goal of such algorithms is to optimize a global function that is the average of local functions using only local computations and communications. Many recent algorithms have been proposed that achieve linear convergence to the global optimum.

We provide a unified analysis that yields the worst-case linear convergence rate as a function of the properties of the functions and underlying network, as well as the parameters of the algorithm. The framework requires solving a small semidefinite program whose feasibility is a sufficient condition for certifying linear convergence of a distributed algorithm. We present results for both known, fixed graphs and unknown, time-varying graphs. The analysis framework is a computationally efficient method for distributed algorithm analysis that enables the rapid comparison, selection, and tuning of algorithms.

This work also makes an effort to systematize distributed algorithm design by devising a canonical form for first-order distributed algorithms. The canonical form characterizes any distributed algorithm that can be implemented using a single round of communication and gradient computation per iteration, and where each agent stores up to two state variables. The canonical form features a minimal set of parameters that are both unique and expressive enough to capture any distributed algorithm in this class. Using this canonical form, we propose a new algorithm, which we call SVL, that is easily implementable and achieves a faster worst-case convergence rate than all other known algorithms.

Chapter 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 $i \in \{1, \ldots, n\}$ has access to a local function $f_i : \mathbb{R}^d \to \mathbb{R}$ and must minimize the average of all agents' local functions

$$\min_{x \in \mathbb{R}^d} f(x), \quad \text{where } f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{1.1}$$

by querying its local gradient ∇f_i, exchanging information with neighboring agents over a graph $\mathcal{G}$, and performing local computations. The objective is for each agent's local memory x_i to eventually converge to $x^\star$, the minimizer of the average of the local functions.

The abstraction above captures a variety of problems that require distributed computation, such as multi-agent coordination and distributed estimation and learning [7,11,16,23]. For example, in large-scale machine learning [7,11], n could represent the number of computing units available for training a large data set. Each f_i then denotes the loss function corresponding to the training examples assigned to unit i. Another example is sensor networks [27], where each sensor may have a limited power budget, communication bandwidth, or sensing capability. The goal is to aggregate all local data without having a single point of failure. Other applications include distributed spectrum sensing [2] and resource allocation across geographic regions [28]. Figure 1.1, which depicts the network capturing the local communication and computation capability of each agent.

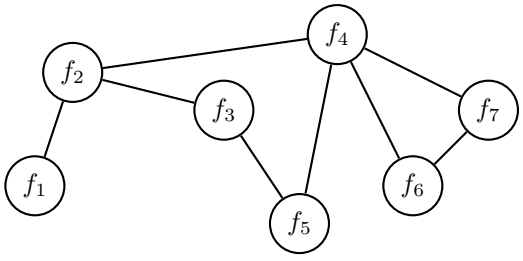

Figure 1.1 Communication graph $\mathcal{G}$ with $n = 7$ nodes. Nodes represent agents with private local functions f_i, and edges represent links across which agents can exchange information with neighbors.

All agents in the network must find the optimal solution $x^\star$ and agree with one another. Distributed optimization thus combines both optimization and average consensus, as we now explain.

Consensus. If each agent uses the initial value x_i^0 and local objective $f_i(x) = \|x - x_i^0\|^2$, distributed optimization reduces to *average consensus* [37, 39]. The unique optimizer of (1.1) is then the average of all initial states: $x^\star = \frac{1}{n}\sum_{i=1}^n x_i^0$. Using a *gossip* update of the form $x_i^{k+1} = \sum_{i=1}^n W_{ij} x_j^k$ where W is carefully chosen, such methods converge exponentially: $\|x_i^k - x^\star\| \leq \rho^k$ with $\rho \in (0,1)$ that depends on W [40]. This is called a *linear rate* in the optimization community. A smaller value of ρ indicates a faster rate of convergence.

Optimization. If $n = 1$ or if all f_i are identical, we recover the standard centralized optimization setup. Linear convergence can be guaranteed in certain cases. For example, the gradient descent method $x_i^{k+1} = x_i^k - \alpha \nabla f_i(x_i^k)$ achieves linear convergence if f_i is continuously differentiable, smooth, and strongly convex [20].

A simple algorithm is *distributed gradient descent* [19], in which agent i uses the update rule:

$$x_i^{k+1} = \sum_{j=1}^n W_{ij} x_j^k - \alpha \nabla f_i(x_i^k). \tag{1.2}$$

Here, x_i^0 is arbitrary and the weights W_{ij} constitute a *gossip matrix* W that is symmetric and doubly stochastic. Moreover, $W_{ij} > 0$ if and only if there is an edge connecting i and j or if $i = j$. The iterations (1.2) combine gradient descent on each f_i with diffusion (consensus) on the x_i. In general, this algorithm requires a diminishing stepsize α in order to converge to $x^\star$ and convergence happens at a sublinear rate even when the f_i are strongly convex. The intuition behind this fact is that the optimal point $x^\star$ is not necessarily a minimizer of the individual f_i, so the agents find themselves taking counterproductive steps.

Since pure consensus achieves linear convergence [39] and so does centralized gradient descent for strongly convex functions [20, 22], one would expect that a combination of consensus and gradient descent could achieve a linear rate as well. Recent efforts have focused on devising such linear-rate algorithms [15, 26, 32].

A linear convergence rate for the general case was first achieved by the *exact first-order algorithm* (EXTRA) [32]. This algorithm requires storing the previous state in memory:

$$x_i^1 = \sum_{j=1}^n W_{ij}\, x_j^0 - \alpha\, \nabla f_i(x_i^0), \qquad x_i^0 \text{ arbitrary}, \tag{1.3a}$$

$$x_i^{k+2} = x_i^{k+1} + \sum_{j=1}^n W_{ij}\, x_j^{k+1} - \sum_{j=1}^n \widetilde{W}_{ij}\, x_j^k - \alpha \left(\nabla f_i(x_i^{k+1}) - \nabla f_i(x_i^k)\right) \tag{1.3b}$$

where W and $\widetilde{W}$ are chosen gossip matrices and α is sufficiently small [32]. Several additional linear-rate algorithms have since been proposed, including: AugDGM [44], DIGing [18, 26], Exact Diffusion [46, 47], NIDS [15], and a unified method [10]. Each of these methods have updates similar to (1.3) in that they require agents to store previous iterates or gradients.

Although linear convergence rates were obtained for the algorithms 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 [42] whereas others provide specific tuning recommendations with associated analytic rate bounds (which may be conservative) [15, 32]. Numerical simulations are also frequently used [41], but can be misleading because algorithm performance depends on the graph topology, choice of functions, algorithm initialization, and algorithm tuning. This dissertation studies the systematic analysis and design of distributed optimization algorithms. We outline the main contributions of this thesis.

Upper Bounds. This work aims to study the reliability of distributed optimization algorithms in two cases: (i) in the presence of a fixed graph, and (ii) in the presence of an unknown *time-varying* communication graph. Such a scenario could occur if communication links fail due to interference, mobile agents move out of range, or an adversary is jamming communications. In both the fixed and time-varying graph cases, we formulate a semidefinite program (SDP) whose solution yields an upper bound on the worst-case linear convergence rate ρ of a wide range of distributed algorithms. The fixed graph result provides graph-dependent results that scale with the size of the network. In the time-varying graph scenario, the SDP is not dependent on the graph and does not scale with n.

Canonical Form. We present a canonical form that characterizes any first-order distributed algorithm that can be implemented using a single round of communication and gradient computation per iteration, and where each agent stores up to two state variables. The canonical form features a minimal set of parameters that are both unique and expressive enough to capture any distributed algorithm in this class. The generic nature of our canonical form enables the systematic analysis and design of distributed optimization algorithms.

Algorithm Design. We present a new distributed algorithm, which we name SVL (the authors' initials). SVL is derived by optimizing the SDP from our analysis framework and provides the fastest known convergence rate to date for the time-varying graph setting. When the graph is well-connected, SVL recovers the performance of gradient descent, which is optimal in the time-varying graph setting.

Lower Bounds. Although our analysis technique only provides *upper bounds* on the worst-case convergence rate for distributed algorithms, we outline a computationally tractable optimization procedure to find numerically matching lower bounds by constructing worst-case trajectories, suggesting that the bounds found via our analysis are tight.

The dissertation is organized as follows. We present our analysis framework for finding upper bounds on worst-case convergence rates for the fixed graph and time-varying graph cases in Chapters 2 and 3, respectively. We derive a canonical form for distributed algorithms in Chapter 4. We present our SVL algorithm and discuss interpretations in Chapter 5. Finally, we demonstrate the tightness of our bounds by generating worst-case trajectories in Chapter 6.

Chapter 2

Analysis over Fixed Graphs

In this chapter, we demonstrate an analysis methodology for distributed optimization algorithms when the graph is known and fixed [34]. The core idea of the analysis framework is to consider distributed algorithms as dynamical systems with feedback. Then, the notion of an algorithm converging corresponds to the feedback control system being stable. Pictorially, Figure 2.1 demonstrates how we represent a distributed algorithm in two parts: a known linear part, which depends on the algorithm parameters (top block in Figure 2.1), and an unknown nonlinear part, characterizing the gradient of the local functions (bottom block in Figure 2.1). We characterize the gradient as the nonlinearity and analyze stability of the closed-loop. Leveraging robust control theory, for all gradient functions in a class of nonlinearities, we would like to guarantee that we obtain a certificate of performance out of the closed-loop map. This worst-case analysis technique is inspired by the seminal work of [13], which analyzes optimization algorithms using integral quadratic constraints. By imposing quadratic constraints on the input and output signals of the nonlinearity, we can obtain a performance certificate for a distributed algorithm. As shown in the model of Figure 2.1, this case involves a single nonlinearity channel corresponding to the gradient of the local functions. We present an analysis approach to prove the linear convergence of EXTRA [11] that depends explicitly on the gossip matrices $W := wij$ and $V := vij$. We will see that this approach can be analogously applied to analyze a variety of other algorithms.

2.1 Preliminaries

We now state a quadratic inequality to characterize the class of sector bounded gradient functions, $\mathcal{F}(m, L)$, so called because functions in this class lie in a sector between lines of slope m and L. For example, m-strongly convex and L-Lipschitz smooth functions are members of $\mathcal{F}(m, L)$.

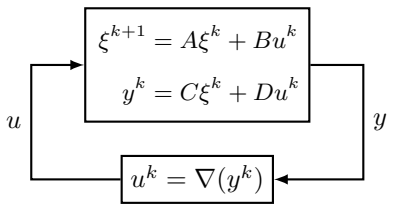

Figure 2.1 Robust analysis over functions.

Proposition 2.1 Suppose $f \in \mathcal{F}(m, L)$, $u^k = \nabla f(y^k)$, and $u^\star = \nabla f(y^\star)$. Then the following inequality holds.

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

Proof. This follows from co-coercivity and Lipschitz property of the gradient. See for example [13, 20]. ■

The following lemma shows that the state of a discrete-time linear dynamical system converges linearly (is exponentially stable, in the language of control theory) provided a certain linear matrix inequality is feasible.

Lemma 2.2 Suppose there exist sequences $\{\xi^k, u^k, y^k\}$ such that for all $k \geq 0$, we have

$$\begin{aligned} \xi^{k+1} &= A\xi^k + Bu^k \\ y^k &= C\xi^k + Du^k \\ 0 &= F\xi^k + Gu^k. \end{aligned} \tag{2.1}$$

where $u^k := \begin{bmatrix} u^{1,k} \\ \vdots \\ u^{p,k} \end{bmatrix}$, $y^k := \begin{bmatrix} y^{1,k} \\ \vdots \\ y^{p,k} \end{bmatrix}$, and (A, B, C, D) is partitioned conformally as

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{c|ccc} A & B^1 & \cdots & B^p \\ \hline C^1 & D^{11} & \cdots & D^{1p} \\ \vdots & \vdots & \ddots & \vdots \\ C^p & D^{p1} & \cdots & D^{pp} \end{array}\right].$$

Also define the block-rows: $D^j := \begin{bmatrix} D^{j1} & \cdots & D^{jp} \end{bmatrix}$. For all $k \geq 0$ and $j = 1, \ldots, p$, further suppose the inputs $u^{j,k}$ and outputs $y^{j,k}$ satisfy the quadratic inequalities

$$\begin{bmatrix} y^{j,k} - y^{j,\star} \\ u^{j,k} - u^{j,\star} \end{bmatrix}^\mathsf{T} M^j \begin{bmatrix} y^{j,k} - y^{j,\star} \\ u^{j,k} - u^{j,\star} \end{bmatrix} \geq 0, \tag{2.2}$$

where $(\xi^\star, y^\star, u^\star)$ is a stationary point of (2.1). Let R be a matrix whose columns are a basis for $\operatorname{null} \begin{bmatrix} F & G \end{bmatrix}$. If there exists $\rho > 0$, $P \succ 0$, and $\lambda_j \geq 0$ such that

$$R^\mathsf{T} \left(\begin{bmatrix} A^\mathsf{T}PA - \rho^2 P & A^\mathsf{T}PB \\ B^\mathsf{T}PA & B^\mathsf{T}PB \end{bmatrix} + \sum_{j=1}^{p} \lambda_j \begin{bmatrix} C^j & D^j \\ 0 & e_j^\mathsf{T} \end{bmatrix}^\mathsf{T} M^j \begin{bmatrix} C^j & D^j \\ 0 & e_j^\mathsf{T} \end{bmatrix} \right) R \preceq 0 \tag{2.3}$$

then $\|\xi^{k+1} - \xi^\star\|_P \leq \rho \|\xi^k - \xi^\star\|_P$ for all $k \geq 0$.

Proof. The columns of R span the nullspace of $\begin{bmatrix} F & G \end{bmatrix}$ so any vector $\begin{bmatrix} (\xi^k - \xi^\star)^\mathsf{T} & (u^k - u^\star)^\mathsf{T} \end{bmatrix}^\mathsf{T}$ is of the form Rw for some w. Multiplying (2.3) on the left and right by w^T and w, respectively, we obtain after simplification:

$$(\xi^{k+1} - \xi^\star)^\mathsf{T} P (\xi^{k+1} - \xi^\star) - \rho^2 (\xi^k - \xi^\star)^\mathsf{T} P (\xi^k - \xi^\star) + \sum_{j=1}^{p} \lambda_j \begin{bmatrix} y^{j,k} - y^\star \\ u^{j,k} - u^\star \end{bmatrix}^\mathsf{T} M^j \begin{bmatrix} y^{j,k} - y^\star \\ u^{j,k} - u^\star \end{bmatrix} \;\leq\; 0.$$

The sum is nonnegative by (2.2), so

$$(\xi^{k+1} - \xi^\star)^\mathsf{T} P (\xi^{k+1} - \xi^\star) \leq \rho^2 (\xi^k - \xi^\star)^\mathsf{T} P (\xi^k - \xi^\star)$$

Taking square roots, the desired result follows. ■

Recursing the result of Lemma 2.2 implies the linear rate bound: $\|\xi^k - \xi^\star\|_P \leq \rho^k \|\xi^0 - \xi^\star\|_P$. We can further bound this via the condition number of P to obtain

$$\|\xi^k - \xi^\star\| \leq \sqrt{\mathrm{cond}(P)}\, \rho^k \, \|\xi^0 - \xi^\star\|$$

If the semidefinite program (2.3) is feasible for some $\rho < 1$, then we have certified a linear convergence rate $O(\rho^k)$. Note that the original bound in Lemma 2.2 is a stronger result because it also provides a *Lyapunov function*, which is a monotonically decreasing function of the state.

2.2 Graph-dependent Analysis Result

In this section, we present an analysis approach to prove the linear convergence of EXTRA [32] that depends explicitly on the gossip matrices $W := \{w_{ij}\}$ and $V := \{v_{ij}\}$.

Theorem 2.3 (W-SDP) Suppose $f_i \in \mathcal{F}(m_i, L_i)$ for $i \in \{1, \ldots, n\}$ and consider the EXTRA algorithm (1.3) with parameter η and gossip matrices W and V. Define the matrices A, B, C, D, F, G as follows.

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] := \left[\begin{array}{ccc|c} W + I_n & -\frac{1}{2}(V + I_n) & \eta I_n & -\eta I_n \\ I_n & 0_n & 0_n & 0_n \\ 0_n & 0_n & 0_n & I_n \\ \hline I_n & 0_n & 0_n & 0_n \end{array}\right],$$

$$F := \begin{bmatrix} 1^\mathsf{T} & -1^\mathsf{T} & \eta 1^\mathsf{T} \end{bmatrix}, \qquad G := 0_{1 \times n}.$$

Further define $\bar{m}$, $\bar{L}$, and M^1 as follows.

$$\bar{m} := \mathrm{diag}(m_1, \ldots, m_n), \quad \bar{L} := \mathrm{diag}(L_1, \ldots, L_n),$$

$$M^1 := \begin{bmatrix} -2\bar{m}\bar{L} & \bar{m} + \bar{L} \\ \bar{m} + \bar{L} & -2I_n \end{bmatrix}.$$

Consider the SDP (2.3) of Lemma 2.2 with $p = 1$ and the matrices A, B, C, D, F, G, M^1 defined as above. If this SDP is feasible for some $\rho > 0$, $P \succ 0$, and $\lambda = 1$, then EXTRA converges linearly with a rate of ρ. In other words, there exists some $c > 0$ such that

$$\|x_i^k - x^\star\| \leq c\,\rho^k \qquad \text{for all } i, k.$$

Proof. Define $\xi^k := \begin{bmatrix}(x^{k+1})^\mathsf{T} & (x^k)^\mathsf{T} & (\nabla^k)^\mathsf{T}\end{bmatrix}^\mathsf{T}$ with

$$x^{k+1} := \begin{bmatrix} x_1^{k+1} \\ \vdots \\ x_n^{k+1} \end{bmatrix}, \quad x^k := \begin{bmatrix} x_1^k \\ \vdots \\ x_n^k \end{bmatrix}, \quad \nabla^k := \begin{bmatrix} \nabla f_1(x_1^k) \\ \vdots \\ \nabla f_n(x_n^k) \end{bmatrix},$$

and input

$$u^{1,k} := \nabla f(y^{1,k}) := \begin{bmatrix} \nabla f_1(y_1^{1,k}) \\ \vdots \\ \nabla f_n(y_n^{1,k}) \end{bmatrix}.$$

In these new coordinates, EXTRA (1.3) takes the form:

$$\begin{aligned} \xi^{k+1} &= (A \otimes I_d)\xi^k + (B \otimes I_d)u^{1,k} \\ y^{1,k} &= (C \otimes I_d)\xi^k + (D \otimes I_d)u^{1,k} \end{aligned}$$

The stationary point of the dynamics is given by $y_i^{1,\star} = x_i^\star = x^\star$, and $u_i^{1,\star} = \nabla f_i(x^\star)$, where $x^\star$ is the global optimum (1.1). Since $f_i \in \mathcal{F}(m_i, L_i)$, the quadratic bound of Proposition 2.1 holds for each agent i. Aggregating the states of all agents we obtain

$$\begin{bmatrix} y^{1,k} - y^{1,\star} \\ u^{1,k} - u^{1,\star} \end{bmatrix}^\mathsf{T} M^1 \begin{bmatrix} y^{1,k} - y^{1,\star} \\ u^{1,k} - u^{1,\star} \end{bmatrix} \geq 0,$$

where M^1 is defined in the theorem statement. Finally, the special initialization condition of EXTRA can also be rewritten as $(F \otimes I_d)\xi^0 = 0$. Moreover,

$$(F \otimes I_d)\xi^{k+1} = (FA \otimes I_d)\xi^k + (FB \otimes I_d)u^k = (F \otimes I_d)\xi^k$$

and it follows that $(F \otimes I_d)\xi^k + (G \otimes I_d)u^k = 0$ for all k. Note that $x^k, \nabla^k, u^{1,k}, y^{1,k} \in \mathbb{R}^{nd}$ and $\xi^k \in \mathbb{R}^{3nd}$. In constructing the SDP (2.3), we may exploit the block-diagonal structure of the algorithm; there will always exist a solution of the form $P \otimes I_d$. See [13, §4.2] for an expanded explanation. Consequently, I_d factors out entirely and we are left with the SDP (2.3) with no dependence on d. By Lemma 2.2, feasibility of (2.3) certifies that $\|\xi^{k+1} - \xi^\star\|_P \leq \rho\|\xi^k - \xi^\star\|_P$. Recursing the bound, we obtain $\|\xi^k - \xi^\star\| \leq \sqrt{\operatorname{cond}(P)}\,\rho^k\,\|\xi^0 - \xi^\star\|$. Note that x_i^k is one of the components of ξ^k and $x^\star$ is the corresponding component of $\xi^\star$. So by the triangle inequality, we have $\|x_i^k - x^\star\| \leq \sqrt{\operatorname{cond} P}\,\rho^k\,\|\xi^0 - \xi^\star\|$, as required. ■

Remark 2.4 When applying Lemma 2.2, the SDP (2.3) is homogeneous in $(P, \lambda_1, \ldots, \lambda_p)$. Therefore, we may set $\lambda_1 = 1$ without loss of generality. This is why $\lambda = 1$ in the statement of Theorem 2.3.

For each fixed $\rho \geq 0$, the SDP (2.3) is a linear matrix inequality (LMI), which is convex and is solved efficiently using interior-point methods or other means. The smallest rate $\rho \geq 0$ for which there exists a feasible $P \succ 0$ may be found using a bisection search. Note that the SDP (2.3) is $4n \times 4n$ with $P \in \mathbb{R}^{3n \times 3n}$. Thus, the size of the SDP is proportional to the number of agents (n), but independent of the size of $x \in \mathbb{R}^d$.

Tightness of Upper Bound. Theorem 2.3 gives an upper bound on the worst case convergence rate. To see whether the bound is tight, we simulate the EXTRA algorithm with random initialization for a two-agent network where each local function is defined by $f_i(x) = \frac{1}{2} x^\mathsf{T} Q_i x - b_i^\mathsf{T} x$. The matrices $Q_i \in \mathbb{R}^{d \times d}$ are symmetric positive semidefinite matrices randomly generated such that $\lambda_{\min}(Q_i) = m$, $\lambda_{\max}(Q_i) = L$, and the rest of the eigenvalues are uniformly distributed in $[m, L]$. Finally, the b_i are random vectors with components independently and uniformly distributed on $[0, 1]$. The gossip matrices used for simulation are $W = V = \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix}$ and both local functions f_1 and f_2 have a condition ratio of $L/m = 10$. The step size parameter used is $\eta = 1/L$. Figure 2.2 depicts several algorithm trajectories upper bounded by the linear rate bound obtained from W-SDP (2.3), which appears tight.

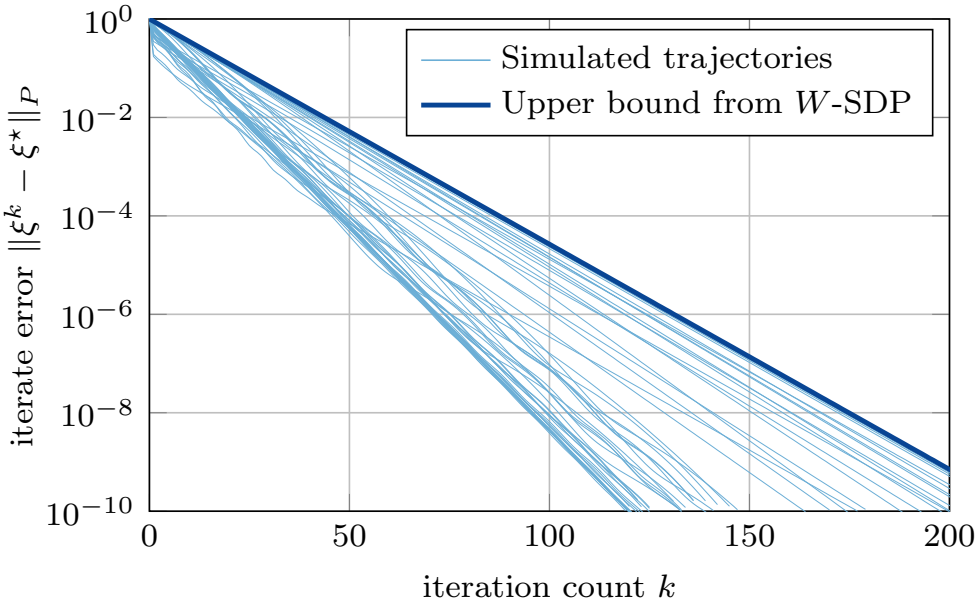

Figure 2.2 Numerical simulations of EXTRA for a network of $n = 2$ agents on 50 randomly generated strongly convex quadratics with $L/m = 10$. The upper bound on the iterate error is found via the W-SDP (2.3).

Varying the Topology. We also experiment with changing the graph topology of the network. For a network with $n = 6$ agents, we consider graphs where each node has degree 5, 4, 3, and 2, respectively. The gossip matrices W and V are chosen to be symmetric and shift-invariant with a second-largest eigenvalue of $\sigma = 2/3$.

In Figure 2.3, we plot the worst-case convergence rate as a function of stepsize η again for the case where $L/m = 10$ for all functions. As the connectivity of the graph grows, EXTRA can tolerate larger stepsizes. The curves overlap and all start similarly, but peel off at different values of η, depending on the graph topology.

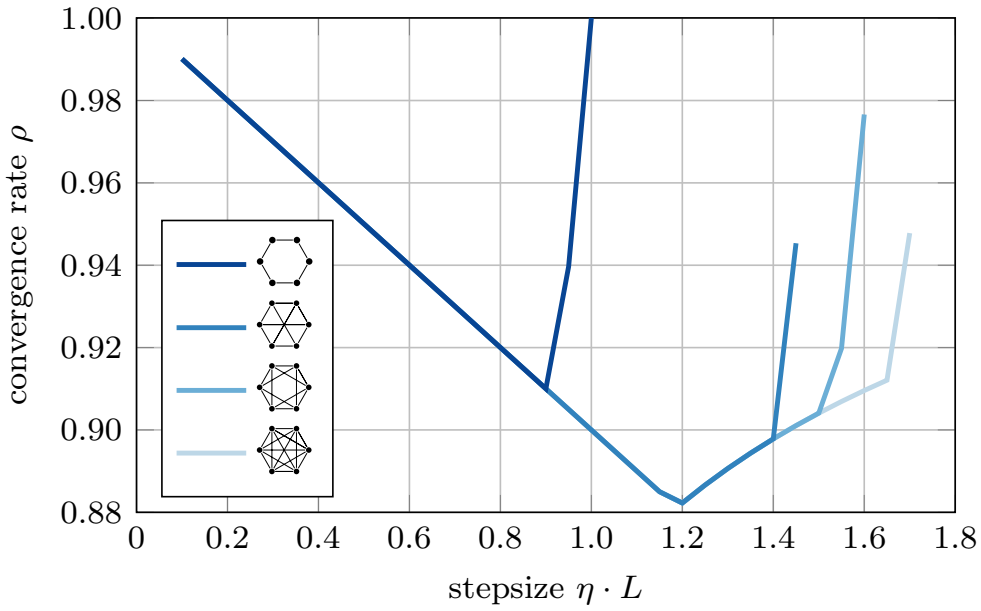

Figure 2.3 Linear rates obtained for EXTRA found via the W-SDP (2.3) as a function of stepsize η for several network topologies with $n = 6$ agents and strongly convex functions with $L/m = 10$. Results suggest that worst-case linear rates are graph-dependent.

Chapter 3

Analysis over Time-Varying Graphs

In this chapter, we consider analysis of distributed algorithms when the graph is changing [36]. W-SDP gives graph-dependent rate bounds, but we would like to obtain robust guarantees even in the time-varying setting. We present a universal analysis framework that provides an upper bound on the worst-case linear convergence rate ρ of a wide range of distributed algorithms as a function of the parameters κ (local function conditioning) and σ (network connectedness). Our main result, Theorem 3.10, is an SDP parameterized by (κ, σ) 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 and is efficiently solvable. Our SDP yields robust performance guarantees when the graph is allowed to vary (even adversarially) at each iteration. Fig. 3.2 compares the worst-case linear rate ρ certified by the SDP for several different algorithms.

The key insight in this case is that we replace the exact gossip matrix with an uncertainty. Introducing an additional channel of robustness, our worst-case performance guarantees are robust over functions and graphs, even when they are allowed to vary at each iteration. Figure 3.1 represents the dual nonlinearity channels in this analysis: one corresponding to the gradient and another corresponding to the graph Laplacian. We will obtain a certificate of stability for the closed loop system via semidefinite programming.

We begin by assembling the ingredients to formulate a semidefinite program to analyze a distributed optimization algorithm: a function class, a graph class, an algorithm, and a fixed point.

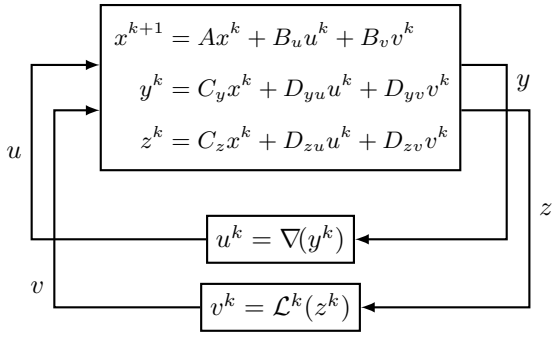

Figure 3.1 Robust analysis over functions and graphs.

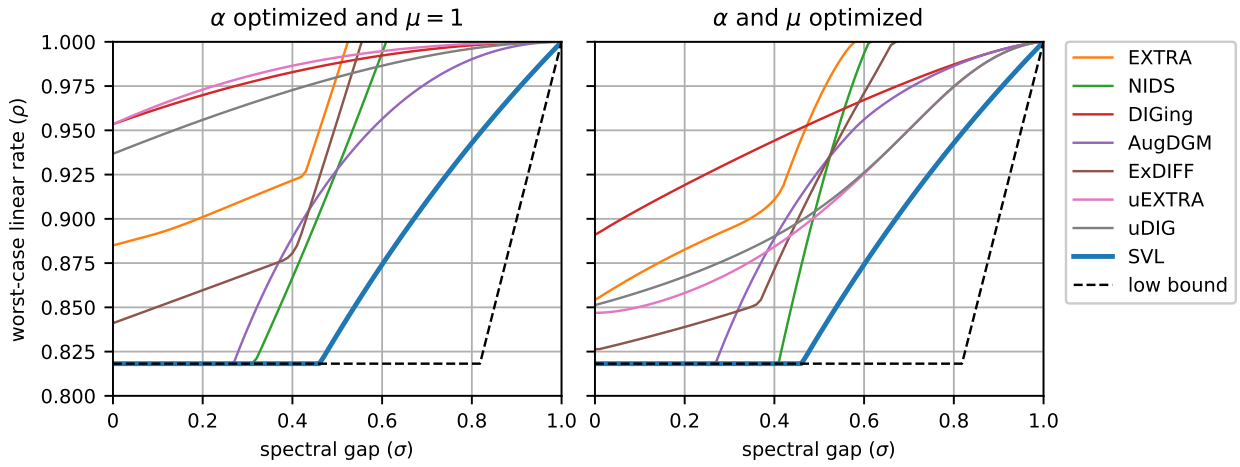

Figure 3.2 Comparison of upper bounds for linear convergence rate ρ (smaller is better) as a function of graph connectedness σ, derived from Theorem 3.10 using $\kappa = 10$. (Left) stepsize α is optimized for each algorithm. (Right) both stepsize α and overelaxation parameter μ are optimized for each algorithm. The SVL algorithm (derived in Chapter 5) outperforms all the tested methods. SVL has no tunable parameters so it is the same in both scenarios. The lower bound (see Chapter 6) corresponds to $\rho \geq \frac{\kappa-1}{\kappa+1} \approx 0.818$ (optimal centralized gradient rate) and $\rho \geq \sigma$ (optimal average consensus rate).

Notation. Unless otherwise indicated, subscripts refer to individual agents while superscripts refer to iteration count. For brevity, we write the symmetric quadratic form $x^{\mathsf{T}}Qx$ as $\begin{bmatrix}\star\end{bmatrix}^{\mathsf{T}} Qx$.

We denote a symbol on agent i at iteration k by x_i^k along with its associated fixed point $x_i^\star$. For all such symbols, we denote their aggregation over all agents as

$$x^k := \begin{bmatrix} x_1^k \\ \vdots \\ x_n^k \end{bmatrix} \quad \text{and} \quad x^\star = \begin{bmatrix} x_1^\star \\ \vdots \\ x_n^\star \end{bmatrix}.$$

We denote the associated local and global error coordinates as $\tilde{x}_i^k := x_i^k - x_i^\star$ and $\tilde{x}^k := x^k - x^\star$, respectively.

3.1 Function and Graph Assumptions

We assume that the local function gradients satisfy the following *sector bound*.

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

$$\big(\nabla f_i(y) - \nabla f_i(y_{\text{opt}}) - m\,(y - y_{\text{opt}})\big)^{\mathsf{T}} \big(\nabla f_i(y) - \nabla f_i(y_{\text{opt}}) - L\,(y - y_{\text{opt}})\big) \leq 0$$

for all $y \in \mathbb{R}^d$, where y_{opt} satisfies $\sum_{i=1}^n \nabla f_i(y_{\text{opt}}) = 0$.

Remark 3.1 One way to satisfy Assumption 1 is if the local functions f_i are m-strongly convex with L-Lipschitz continuous gradients, though in general Assumption 1 is weaker.

We define the condition ratio as $\kappa := L/m$. This quantity captures the amount of variation in the curvature of the objective function. If f_i is twice differentiable, κ is an upper bound on the condition number of the Hessian $\nabla^2 f_i$. In general, as $\kappa \to \infty$, the functions become poorly conditioned and more difficult to optimize using first-order methods.

Define the graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} := \{1, \dots, n\}$ 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 $\mathcal{L}1_n = 0$ 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 matrix associated with $\mathcal{G}^k$. The graph associated with the network of agents can in general change at each time step k of the algorithm, so we assume the following 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: 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: every node has equal in-degree and out-degree. This means that $1_n^\mathsf{T} \mathcal{L}^k = 0$ for all k.
3. The spectral gap of the time-varying graph is uniformly bounded. In particular, we assume there exists $\sigma \in [0, 1)$ such that $\|I - \Pi - \mathcal{L}^k\| \leq \sigma$ 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 $\{\mathcal{L}^k\}$.

Remark 3.2 The assumption that $\mathcal{G}^k$ must be connected for all k is a strong assumption. Works that consider directed or time varying graphs typically make weaker assumptions, such as a *joint spectrum property* or B*-connectedness* [18]. Nevertheless, our setting (which is equivalent to B-connectedness with $B = 1$) is still weaker than assuming a constant graph. Indeed, NIDS [15] converges for any σ when the graph is constant, but in Chapter 6, we construct a sequence of graphs that drives NIDS to instability.

3.2 Algorithm Form

We consider the broad class of distributed optimization algorithms that satisfy the algebraic equations

$$\begin{bmatrix} x_i^{k+1} \\ y_i^k \\ z_i^k \end{bmatrix} = \begin{bmatrix} A & B_u & B_v \\ C_y & D_{yu} & D_{yv} \\ C_z & D_{zu} & D_{zv} \end{bmatrix} \begin{bmatrix} x_i^k \\ u_i^k \\ v_i^k \end{bmatrix}, \tag{3.1a}$$

$$u_i^k = \nabla f_i(y_i^k), \qquad v_i^k = \sum_{j=1}^n \mathcal{L}_{ij}^k z_j^k, \tag{3.1b}$$

$$\sum_{j=1}^n \left(F_x x_j^k + F_u u_j^k\right) = 0. \tag{3.1c}$$

Equation (3.1a) describes how agent i's state x_i^k evolves with iteration k. The local gradient ∇f_i is evaluated at y_i^k and the quantity z_i^k is transmitted to neighboring agents in (3.1b). Finally, we allow for linear state-input invariants to be enforced in (3.1c). Such invariants typically arise from requiring a particular initialization for the algorithm.

The matrices A, D_{yu}, and D_{zv} are square, and the other matrices have compatible dimensions. The dimension of A is the number of local states on each agent, the dimension of D_{yu} is one, and the dimension of D_{zv} is the number of variables that each agent transmits with neighbors at each iteration.

Remark 3.3 (Dimension Reduction) To simplify notation, we assume the objective function is one-dimensional ($d = 1$). We can recover the general d case by replacing each scalar symbol with a $1 \times d$ row vector (e.g., $u_i^k \in \mathbb{R}^{1\times d}$) and interpreting each local gradient ∇f_i as a map from $\mathbb{R}^{1\times d}$ to $\mathbb{R}^{1\times d}$.

Remark 3.4 (Implementation) Not all instances of (3.1) are efficiently implementable. For example, if $D_{yu} \neq 0$, then y_i^k depends on u_i^k, which then depends on y_i^k. Such circular dependencies arise naturally in proximal algorithms, where an inner optimization problem must be solved at each iteration. For instance, given a convex differentiable f and parameter $\lambda > 0$, the proximal algorithm

$$x^{k+1} = \mathbf{prox}_{\lambda f}(x^k) := \arg\min_x \left(\lambda f(x) + \tfrac{1}{2}\|x - x^k\|^2\right)$$

satisfies the optimality condition $\lambda \nabla f(x^{k+1}) + x^{k+1} - x^k = 0$ and can therefore be expressed in the form of (3.1) as follows:

$$x^{k+1} = x^k - \lambda u^k, \qquad y^k = x^k - \lambda u^k, \qquad u^k = \nabla f(y^k).$$

In the forthcoming analysis, we treat implementability and analysis separately. That is, we derive convergence rate bounds for general algorithms of the form (3.1), regardless of whether they can be efficiently implemented. However, we note that a sufficient condition for avoiding circular dependencies is if the feedthrough term satisfies

$$\begin{bmatrix} D_{yu} & D_{yv} \\ D_{zu} & D_{zv} \end{bmatrix} = \begin{bmatrix} 0 & D_{yv} \\ 0 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 0 & 0 \\ D_{zu} & 0 \end{bmatrix}. \tag{3.2}$$

Putting a distributed optimization algorithm into the form of (3.1) is a straightforward algebraic exercise, which we now demonstrate for two recently proposed algorithms. These algorithms are parameterized by a stepsize α and a gossip matrix W. To relate the gossip matrix to the Laplacian matrix, we set $W = I - \mu\mathcal{L}$ for some scalar $\mu \neq 0$. This provides an additional tuning parameter, and is akin to the method of successive overrelaxation used in the numerical solutions of linear systems of equations [21].

EXTRA. The EXTRA algorithm (1.3) has a state that depends on two previous timesteps. Using the authors' recommendation of $\widetilde{W} = \frac{1}{2}(I + W)$ together with $W = I - \mu\mathcal{L}^k$, the equations become

$$\begin{aligned} x^1 &= x^0 - \alpha\nabla f(x^0) - \mu\mathcal{L}^k x^0, \\ x^{k+2} &= 2x^{k+1} - x^k - \alpha\left(\nabla f(x^{k+1}) - \nabla f(x^k)\right) - \mu\mathcal{L}^k\left(x^{k+1} - \tfrac{1}{2}x^k\right). \end{aligned}$$

Define the state $(x^{k+1}, x^k, \nabla f(x^k))$. The outputs are now functions of the state: $y^k := x^{k+1}$ and $z^k := x^{k+1} - \frac{1}{2}x^k$. Finally, summing across agents (left-multiplying by 1^T) and using $1^\mathsf{T}\mathcal{L}^k = 0$, we find that $1^\mathsf{T}\left(x^{k+1} - x^k + \alpha\nabla f(x^k)\right)$ is independent of k, and identically zero due to how x^1 is initialized. The parameters that characterize EXTRA are shown below and in Table 3.1.

$$\left[\begin{array}{c|c:c} A & B_u & B_v \\ \hdashline C_y & D_{yu} & D_{yv} \\ \hdashline C_z & D_{zu} & D_{zv} \\ \hdashline F_x & F_u & \end{array}\right] = \left[\begin{array}{ccc:cc} 2 & -1 & \alpha & -\alpha & -\mu \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & -\frac{1}{2} & 0 & 0 & 0 \\ \hdashline 1 & -1 & \alpha & 0 & \end{array}\right].$$

DIGing. The DIGing algorithm [18, 26], is an example of a *gradient tracking* algorithm. It begins with an arbitrary initialization x^0 and has two update equations, as follows:

$$\begin{aligned} s^0 &= \nabla f(x^0), \\ x^{k+1} &= Wx^k - \alpha s^k, \\ s^{k+1} &= \widetilde{W}s^k + \nabla f(x^{k+1}) - \nabla f(x^k). \end{aligned}$$

Using the authors' recommendation of $\widetilde{W} = W$, defining $W = I - \mu\mathcal{L}^k$ as before, and defining the state as $(x^k, s^k, \nabla f(x^k))$, we find that the output is $y^k := x^{k+1}$, two quantities must be communicated between agents, $z^k := (x^k, s^k)$, and the invariant is $1^\mathsf{T}(s^k - \nabla f(x^k)) = 0$. The parameters that characterize DIGing are shown in Table 3.1.

A similar derivation can be applied to a variety of algorithms. Table 3.1 summarizes the parameterizations for several recently proposed algorithms.

3.3 Fixed Points

Before studying the performance of distributed algorithms, we must establish the notion of a fixed point. This is essential for analysis as the existence of a fixed point is a prerequisite for proving convergence. Unlike the centralized setting, each agent in the network must have a fixed point. The added subtlety is that the aggregate fixed point for all agents must solve the necessary conditions for optimality and consensus. That is, at the fixed point, the sum of the local gradients must be zero and all agents' local variables must agree. In this section, we derive conditions for a distributed algorithm to have a fixed point. For an algorithm to be valid, (i) there must exist a fixed point corresponding to the optimal solution, and (ii) the iterates must converge to the fixed point. We explore the topic of convergence to the fixed point in the next chapter.

In the case of a time-varying graph, the fixed point must be shown to solve the optimization problem (1.1) even when the Laplacian matrix changes at each step. Considering the dynamical system that characterizes a distributed algorithm, we study the associated fixed-point equations and develop conditions for the existence of a solution.

KKT Conditions. First, we consider an equivalent formulation of the distributed optimization problem with the consensus constraint expressed explicitly.

$$\begin{aligned} y_{\text{opt}} = \underset{y_i \in \mathbb{R}^d}{\arg\min} \quad & \frac{1}{n} \sum_{i=1}^{n} f_i(y_i) \\ \text{s.t.} \quad & y_i = y_j \quad \forall i, j \end{aligned} \tag{3.3}$$

We state the necessary conditions for optimality and consensus using the KKT conditions for primal and dual feasibility of (3.3).

$$\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(y_{\text{opt}}) = 0 \quad \text{and} \quad y_i^\star = y_{\text{opt}} \; \forall \, i.$$

The above conditions must hold for an optimal fixed point of any distributed algorithm.

We provide simple conditions for verifying the existence of a fixed point for distributed algorithms over a time-varying graph. Consider the fixed point equations associated with (3.1)

$$\begin{aligned} x_i^\star &= A x_i^\star + B_u u_i^\star + B_v v_i^\star \\ y_i^\star &= C x_i^\star + D_{yu} u_i^\star + D_{yv} v_i^\star \\ z_i^\star &= C_z x_i^\star + D_{zu} u_i^\star + D_{zv} v_i^\star \end{aligned} \tag{3.4a}$$

$$u_i^\star = \nabla f_i(y_i^\star), \qquad v_i^\star = \sum_{j=1}^{n} \mathcal{L}_{ij}^k z_j^\star, \tag{3.4b}$$

$$\sum_{j=1}^{n} \left(F_x x_j^\star + F_u u_j^\star \right) = 0. \tag{3.4c}$$

Not all instances of algorithm (3.1) solve the distributed optimization problem (1.1). We seek a fixed point of (3.1), which means there exists a solution $(x_i, y_i, z_i, u_i, v_i)$ to (3.4). In addition, the fixed point must fulfill certain properties to be valid. Considering the aggregate variables for all agents, a distributed algorithm of the form (3.1) has a fixed point $(x^\star, y^\star, z^\star, u^\star, v^\star)$ corresponding to the optimal solution of (1.1) for all sector-bounded functions satisfying Assumption 1, and all graphs satisfying Assumption 2, if the following conditions hold.

Fixed Point Conditions

Consensus. All agents must achieve consensus on the point at which the gradient is evaluated. This means that the fixed point must satisfy $y_1^\star = \ldots = y_n^\star$.

$$(I - \Pi)\, y^\star = 0. \tag{3.5}$$

Optimality. The point must be a stationary (first-order optimal) point of f. and $u_1^\star + \cdots + u_n^\star = 0$, or in vector form,

$$1^\mathsf{T} u^\star = 0. \tag{3.6}$$

Robustness to Graph. The fixed point must not depend on the sequence of graphs $\{\mathcal{L}^k\}$, so $z_1^\star = \ldots = z_n^\star$ and $v_1^\star = \cdots = v_n^\star = 0$, or in vector form,

$$(I - \Pi)\, z^\star = 0 \quad \text{and} \quad v^\star = 0. \tag{3.7}$$

Robustness Functions. The fixed point must satisfy $y_1^\star = \ldots = y_n^\star = y_{\text{opt}}$ and $u_i^\star = \nabla f_i(y_{\text{opt}})$, where y_{opt} is the optimizer of (1.1). For these to hold for any objective function f, we need

$$1^\mathsf{T} y^\star \text{ and } (I - \Pi)\, u^\star \text{ unconstrained.} \tag{3.8}$$

The following proposition characterizes algorithms with such a fixed point.

Proposition 3.5 (Existence of Fixed Point) An algorithm of the form (3.1) has a fixed point $(x^\star, y^\star, z^\star, u^\star, v^\star)$ that satisfies the conditions in (3.5)–(3.8) if and only if

$$\text{null}(A - I) \cap \text{row}(C_y) \cap \text{null}(F_x) \neq \{0\} \tag{3.9a}$$

$$\text{and} \quad \begin{bmatrix} B_u \\ D_{yu} \\ D_{zu} \end{bmatrix} \in \text{col}\left(\begin{bmatrix} A - I \\ C_y \\ C_z \end{bmatrix} \right). \tag{3.9b}$$

Here, "null", "col", and "row" denote the nullspace, column space, and row space, respectively. Both EXTRA and DIGing as derived above satisfy the conditions in (3.9) and therefore have a fixed point corresponding to the optimal solution of (1.1).

Proof. Suppose (3.9) holds, and denote the optimizer of (1.1) by y_{opt}. Then there exist vectors p and q such that

$$\begin{cases} 0 = (A - I)\, p \\ y_{\text{opt}} = C_y p \\ 0 = F_x p \end{cases} \quad \text{and} \quad \begin{cases} B_u = (A - I)\, q \\ D_{yu} = C_y q \\ D_{zu} = C_z q. \end{cases}$$

For all $i \in \{1, \ldots, n\}$, use these vectors to define the points

$$\begin{aligned} x_i^\star &= p - q\, \nabla f_i(y_{\text{opt}}), & y_i^\star &= y_{\text{opt}}, & z_i^\star &= C_z p, \\ u_i^\star &= \nabla f_i(y_{\text{opt}}), & v_i^\star &= 0. \end{aligned}$$

This is a fixed point of algorithm (3.1), and the fixed point satisfies the conditions in (3.5)–(3.8) since y_{opt} is the optimizer of (1.1).

Now suppose $(x^\star, y^\star, z^\star, u^\star, v^\star)$ is a fixed point of (3.1) satisfying (3.5)–(3.8). Let $p = (1/n) \sum_{i=1}^n x_i^\star$. Since $1^\mathsf{T} u^\star = 0$, $v^\star = 0$, and $1^\mathsf{T} y^\star$ is unconstrained, we have from (3.1a) and (3.1c) that $p \neq 0$ is in the set (3.9a). Now let r be any nonzero vector such that $r^\mathsf{T} 1 = 0$. Then from (3.1a), we have that

$$0 = \begin{bmatrix} A - I \\ C_y \\ C_z \end{bmatrix} (r^\mathsf{T} x^\star) + \begin{bmatrix} B_u \\ D_{yu} \\ D_{zu} \end{bmatrix} (r^\mathsf{T} u^\star).$$

Since this must hold for arbitrary $r^\mathsf{T} u^\star$, this implies (3.9b). ■

Remark 3.6 Proposition 3.5 guarantees that any instance of algorithm (3.1) satisfying (3.5)–(3.8) has a desirable fixed point in the presence of a time-varying graph; all agents agree on a common stationary point of (1.1). However, Proposition 3.5 does not ensure that the algorithm necessarily converges to this fixed point, nor does it characterize the rate of convergence. These questions will be explored in Section 3.5.

Remark 3.7 DIGing has a fixed point in the case of a time varying graph. Equation (3.4) depends on the Laplacian at each step, so a fixed point in general does not exist because it would depend on the graph at each step. This is why we require the additional condition of robustness to graphs for the existence of a fixed point in the case of DIGing and other algorithms that communicate two variables.

3.4 Lower Bounds on Worst-Case Convergence Rates

We now construct simple lower bounds on the worst-case asymptotic convergence rate of the iterates for any valid algorithm of the form (3.1). We do so by separately considering the two specific instances discussed in Chapter 1.

Consensus. Consider the scalar local quadratic functions $f_i(y) = \frac{L}{2}(y - r_i)^2$. Then Assumption 1 holds with $m = L$ and $y_{\text{opt}} = \frac{1}{n} \sum_{i=1}^n r_i$.

Optimization. Consider the case $n = 1$. For the graph to satisfy Assumption 2, the Laplacian matrix must be $\mathcal{L}^k = 0$, which has spectral gap $\sigma = 0$.

In both cases above, the algorithm reduces to a linear system in feedback with sector-bounded nonlinearity: in the sector $(1-\sigma, 1+\sigma)$ for consensus and (m, L) for optimization. Further, the linear part of the system is strictly proper (since the algorithm is implementable) and must contain an integrator (due to the fixed-point conditions). Then, using the lower bound for such systems in [14], we obtain the following.

Proposition 3.8 For any $\rho < \max\{\frac{\kappa-1}{\kappa+1}, \sigma\}$, there does *not* exist an algorithm of the form (3.1) that satisfies the implementability conditions (3.2) and fixed-point conditions (3.9) such that, for all objective functions and Laplacian matrices satisfying Assumptions 1 and 2, there exists a constant $c > 0$ such that the bound $\|x_i^k - y_{\text{opt}}\| \leq c\,\rho^k$ holds for all agents $i \in \{1, \ldots, n\}$ and all iterations $k \geq 0$.

Remark 3.9 (Accelerated Rates) Distributed algorithms that achieve *accelerated* [25, 41, 43] or *optimal* [30] linear rates have also been proposed. It turns out such methods are not guaranteed to achieve acceleration when the graph is time-varying. These lower bounds, which are achieved by ordinary gradient descent, imply that accelerated algorithms such as the recently proposed SSDA [30], or distributed versions of heavy-ball [43] or Nesterov acceleration [25, 41], do not in fact achieve accelerated rates in the worst case when the local function gradients are sector bounded and the graph is time-varying, as in Assumptions 1 and 2, respectively.

3.5 Graph-independent Analysis Result

The main result of this chapter follows. The semidefinite program (3.10) in Theorem 3.10 poses a sufficient condition for convergence. That is, a feasible solution yields a stability certificate that the algorithm converges at rate ρ in the *worst case*. The SDP enjoys scale-invariance: it neither grows with function dimension d nor with size of the network n (the number of agents). We use solvers to solve the SDP (3.10) numerically. To find the smallest rate ρ, we perform bisection search.

Theorem 3.10 (Analysis Result) Consider the distributed optimization problem (1.1) solved using algorithm (3.1). Suppose Assumptions 1 and 2 hold and further assume the algorithm satisfies the fixed point conditions (3.9). Define the matrices

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

Let Ψ be a matrix whose columns form a basis for the nullspace of $\begin{bmatrix} F_x & F_u \end{bmatrix}$. If there exist $P \succ 0$, $Q \succ 0$, and $R \succeq 0$ of appropriate sizes such that

$$\Psi^{\mathsf{T}} \begin{bmatrix} \star \end{bmatrix}^{\mathsf{T}} \left[\begin{array}{cc:c} P & 0 & 0 \\ 0 & -\rho^2 P & 0 \\ \hdashline 0 & 0 & M_0 \end{array}\right] \left[\begin{array}{cc} A & B_u \\ I & 0 \\ \hdashline C_y & D_{yu} \\ 0 & I \end{array}\right] \Psi \preceq 0 \tag{3.10a}$$

$$\begin{bmatrix} \star \end{bmatrix}^{\mathsf{T}} \left[\begin{array}{cc:c:c} Q & 0 & 0 & 0 \\ 0 & -\rho^2 Q & 0 & 0 \\ \hdashline 0 & 0 & M_0 & 0 \\ \hdashline 0 & 0 & 0 & M_1 \otimes R \end{array}\right] \left[\begin{array}{ccc} A & B_u & B_v \\ I & 0 & 0 \\ \hdashline C_y & D_{yu} & D_{yv} \\ 0 & I & 0 \\ \hdashline C_z & D_{zu} & D_{zv} \\ 0 & 0 & I \end{array}\right] \preceq 0 \tag{3.10b}$$

then there exists a constant[1] $c > 0$ independent of i and k such that for all agents $i \in \{1, \dots, n\}$ and all iterations $k \geq 0$,

$$\|x_i^k - x_i^\star\| \leq c\,\rho^k \tag{3.11}$$

for some fixed point $(x_i^\star, y_i^\star, z_i^\star, u_i^\star, v_i^\star)$ that satisfies (3.5)–(3.8).

For fixed algorithm parameters $A, B_u, B_v, C_y, C_z, D_{yu}$, $D_{yv}, D_{zu}, D_{zv}, F_x, F_u$, function parameters m and L, graph parameter σ, and candidate rate ρ, the SDP (3.10) is a linear matrix inequality (LMI) in the variables (P, Q, R), and therefore convex. Indeed, (3.10a) and (3.10b) are decoupled and their feasibility may be checked separately. To find the best (smallest) upper bound, we observe that feasibility of (3.10) 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 (3.10) is not jointly convex in (P, Q, R, ρ). While our result is only a sufficient condition for convergence, we provide empirical evidence in Chapter 6 that suggests that it is in fact tight.

Remark 3.11 Our main theorem provides conditions under which the *state* converges to a fixed point linearly with rate ρ. However, when the algorithm also satisfies the conditions in (3.2) for being efficiently implementable, then under the conditions of Theorem 3.10, there exist constants c_u, c_v, c_y, and c_z such that for all agents i and all iterations k,

$$\|u_i^k - u_i^\star\| \leq c_u\,\rho^k, \qquad \|y_i^k - y_i^\star\| \leq c_y\,\rho^k,$$
$$\|v_i^k - v_i^\star\| \leq c_v\,\rho^k, \qquad \|z_i^k - z_i^\star\| \leq c_z\,\rho^k,$$

[1] The constant c does not depend on the particular local functions f_i and sequence $\{\mathcal{L}^k\}$, but does depend on $x_i^0 - x_i^\star$. We provide an explicit formula for c in the proof in Section 3.5.

for some fixed point $(x_i^\star, y_i^\star, z_i^\star, u_i^\star, v_i^\star)$ that satisfies (3.5)–(3.8). In particular, the output sequence y_i^k of each agent converges to the optimizer y_{opt} of (1.1) linearly with rate ρ.

The core idea behind Theorem 3.10 is to posit a quadratic Lyapunov candidate of the form

$$V^k := (x^k - x^\star)^\mathsf{T}\big(\Pi \otimes P + (I - \Pi) \otimes Q\big)(x^k - x^\star) \tag{3.12}$$

for some appropriate choice of $P \succ 0$ and $Q \succ 0$. Feasibility of (3.10) 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 (and less concise) version of Theorem 3.10 appeared in [34].

Algorithm Comparison. Theorem 3.10 provides an upper bound on the worst-case convergence rate. We used this result to compare all algorithms in Table 3.1, including SVL. The results are shown in Fig. 3.2.

For each algorithm, we used a bisection search to find the smallest rate ρ that yielded a feasible solution to the SDP (3.10). We implemented the SDP in Julia [3] with the JuMP [6] modeling package and the Mosek interior point solver [1]. In an outer loop, we performed a parameter search for each algorithm to find the step size α and overrelaxation parameter μ that yielded the smallest possible ρ. Specifically, we used Brent's method and the Nelder–Mead method, respectively, as implemented in the Optim package [17] as σ ranged from 0 to 1.

As shown in Fig. 3.2, optimizing over μ further improves worst-case performance. Our proposed SVL algorithm outperforms all methods we tested. Also shown in Fig. 3.2 is the lower bound described in Section 3.4, namely $\rho \geq \max\{\frac{\kappa-1}{\kappa+1}, \sigma\}$, which holds for any distributed algorithm.

Proof of Main Analysis Result

The key idea of the main result is to find a Lyapunov function that decreases exponentially along all trajectories of the algorithm. Two LMIs emerge: one from the *consensus* direction, corresponding to the all-ones eigenvector of the Laplacian, and another from the *disagreement* direction, the direction orthogonal to the all-ones vector. 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 3.12 Suppose Assumption 1 holds for the local objective functions f_i. Let (y_i^k, u_i^k) satisfy (3.1b), and let $(y_i^\star, u_i^\star)$ be a fixed point that satisfies (3.5)–(3.8). Then

$$\begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^\mathsf{T} (M_0 \otimes I) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} \geq 0. \tag{3.13}$$

Proof. Using the definition of M_0, the quadratic form is

$$\begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^\mathsf{T} (M_0 \otimes I) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} = -2 \sum_{i=1}^n (\tilde{u}_i^k - m\tilde{y}_i^k)^\mathsf{T} (\tilde{u}_i^k - L\tilde{y}_i^k).$$

Since the fixed point satisfies (3.5)–(3.8), Assumption 1 implies that this is nonnegative with $y_{\text{opt}} = y_1^\star = \ldots = y_n^\star$. ■

Proposition 3.13 Suppose Assumption 2 holds for the graph $\mathcal{G}^k$ at each iteration. Let (z_i^k, v_i^k) satisfy (3.1b), and let $(z_i^\star, v_i^\star)$ be a fixed point that satisfies (3.5)–(3.8). Then for all $R \succeq 0$,

$$\begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix}^{\mathsf{T}} \big(M_1 \otimes (I - \Pi) \otimes R\big) \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix} \geq 0. \tag{3.14}$$

A proof of Proposition 3.13 is provided in Appendix A.1.

Let $(x^k, y^k, z^k, u^k, v^k)$ denote a trajectory of algorithm (3.1). Since the algorithm satisfies the fixed point conditions (3.9) (by assumption), we have from Proposition 3.5 that there exists a fixed point $(x^\star, y^\star, z^\star, u^\star, v^\star)$ satisfying (3.5)–(3.8). The global optimizer is unique from Assumption 1, so the fixed point conditions (3.5) imply that $y_1^\star = \ldots = y_n^\star = y_{\text{opt}}$ with y_{opt} the optimizer of (1.1).

Since the trajectory satisfies the invariant (3.1c) and the columns of Ψ form a basis for the nullspace of $\begin{bmatrix} F_x & F_u \end{bmatrix}$, there exists a vector $\tilde{s}^k$ such that

$$\Psi\, \tilde{s}^k = \frac{1}{\sqrt{n}} \sum_{i=1}^n \begin{bmatrix} \tilde{x}_i^k \\ \tilde{u}_i^k \end{bmatrix}.$$

Multiplying the matrix in (3.10a) on the right and left by $\tilde{s}^k$ and its transpose, respectively, we obtain the consensus inequality

$$(\tilde{x}^{k+1})^{\mathsf{T}} (\Pi \otimes P)\, \tilde{x}^{k+1} - \rho^2\, (\tilde{x}^k)^{\mathsf{T}} (\Pi \otimes P)\, \tilde{x}^k + \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^{\mathsf{T}} (M_0 \otimes \Pi) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} \leq 0. \tag{3.15a}$$

Now choose the vectors $w_2, \ldots, w_n \in \mathbb{R}^n$ such that the matrix $\begin{bmatrix} 1_n/\sqrt{n} & w_2 & \ldots & w_n \end{bmatrix}$ is orthonormal. Then we can multiply the matrix in (3.10b) on the right and left by the weighted sum

$$\sum_{i=1}^n (w_\ell)_i \begin{bmatrix} \tilde{x}_i^k \\ \tilde{u}_i^k \\ \tilde{v}_i^k \end{bmatrix}$$

and its transpose, respectively, and sum over $\ell \in \{2, \ldots, n\}$ to obtain the disagreement inequality

$$\begin{aligned}(\tilde{x}^{k+1})^{\mathsf{T}} \big((I - \Pi) \otimes Q\big) \tilde{x}^{k+1} - \rho^2 (\tilde{x}^k)^{\mathsf{T}} \big((I - \Pi) \otimes Q\big) \tilde{x}^k \\ + \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^{\mathsf{T}} \big(M_0 \otimes (I - \Pi)\big) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} + \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix}^{\mathsf{T}} \big(M_1 \otimes (I - \Pi) \otimes R\big) \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix} \leq 0,\end{aligned} \tag{3.15b}$$

where we used that $\{w_i\}_{i=1}^n$ form an orthonormal basis for $\mathbb{R}^n$. Summing the inequalities in (3.15), we obtain

$$V^{k+1} - \rho^2 V^k + \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^{\mathsf{T}} (M_0 \otimes I) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} + \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix}^{\mathsf{T}} \big(M_1 \otimes (I - \Pi) \otimes R\big) \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix} \leq 0,$$

where V^k is defined in (3.12). The quadratic forms in the last two terms are nonnegative from Propositions 3.12 and 3.13, which implies $V^{k+1} \leq \rho^2 V^k$. We then apply this inequality iteratively to obtain $V^k \leq \rho^{2k} V_0$ for all $k \geq 0$. Now define

$$T := \Pi \otimes P + (I - \Pi) \otimes Q,$$

and note that $T \succ 0$ since P and Q are positive definite. Then letting $\mathrm{cond}(T) = \lambda_{\max}(T)/\lambda_{\min}(T)$ denote the condition number of T, we have the bound

$$\|x_i^k - x_i^\star\|^2 \leq \|x^k - x^\star\|^2 \leq \mathrm{cond}(T)\, V^k \leq \rho^{2k}\, \mathrm{cond}(T)\, V^0.$$

Therefore, the bound (3.11) holds with $c = \sqrt{\mathrm{cond}(T)\, V^0}$. ■

Table 3.1 Algorithm parameters in the form of (3.1) for a variety of different distributed optimization algorithms. Algorithms can be tuned by choosing stepsize and overrelaxation parameters α and μ, respectively. Algorithms are organized based on how many internal states they have (columns) and how many variables must be communicated in each iteration (block rows).

	Algorithms with 2 states		Algorithms with 3 states	
1 communicated variable	**SVL template (present work)** See Chapter 5 for derivation of $(\alpha, \beta, \gamma, \delta)$	$\left[\begin{array}{cc:c:c} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \\ \hdashline 1 & 0 & 0 & -\delta \\ \hdashline 1 & 0 & 0 & 0 \\ \hdashline 0 & 1 & 0 & \end{array}\right]$	EXTRA [32]	$\left[\begin{array}{cc:c:c:c} 2 & -1 & \alpha & -\alpha & -\mu \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & -\frac{1}{2} & 0 & 0 & 0 \\ \hdashline 1 & -1 & \alpha & 0 & \end{array}\right]$
	Exact Diffusion (ExDIFF) [46, 47]	$\left[\begin{array}{cc:c:c} 2 & -1 & -\alpha & -\mu \\ 1 & 0 & -\alpha & -\frac{1}{2}\mu \\ \hdashline 1 & 0 & -\frac{1}{2}\mu & 0 \\ \hdashline 1 & 0 & 0 & 0 \\ \hdashline 1 & -1 & 0 & \end{array}\right]$	NIDS [15]	$\left[\begin{array}{cc:c:c:c} 2 & -1 & \alpha & -\alpha & -\mu \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & -\frac{1}{2} & \frac{\alpha}{2} & -\frac{\alpha}{2} & 0 \\ \hdashline 1 & -1 & \alpha & 0 & \end{array}\right]$
2 communicated variables	Unified DIGing (uDIG) [10]	$\left[\begin{array}{ccc:c:c} 1 & -\alpha & -\alpha & -\mu & 0 \\ 0 & 1 & 0 & 0 & -\mu \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline -\frac{L+m}{2} & 1 & 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 & & \end{array}\right]$	DIGing [18, 26]	$\left[\begin{array}{ccc:c:c:c} 1 & -\alpha & 0 & 0 & -\mu & 0 \\ 0 & 1 & -1 & 1 & 0 & -\mu \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 1 & -\alpha & 0 & 0 & -\mu & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 1 & -1 & 0 & & \end{array}\right]$
	Unified EXTRA (uEXTRA) [10]	$\left[\begin{array}{ccc:c:c} 1 & -\alpha & -\alpha & -\mu & 0 \\ 0 & 1 & 0 & 0 & -\mu \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline -L & 1 & 1 & L\mu & 0 \\ \hdashline 0 & 1 & 0 & & \end{array}\right]$	AugDGM [44]	$\left[\begin{array}{ccc:c:c:c} 1 & -\alpha & 0 & 0 & -\mu & \alpha\mu \\ 0 & 1 & -1 & 1 & 0 & -\mu \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 1 & -\alpha & 0 & 0 & -\mu & \alpha\mu \\ \hdashline 1 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 1 & -1 & 0 & & \end{array}\right]$

Chapter 4

Canonical Form

In this chapter we take a first step towards systematic design of distributed algorithms, featuring the work of [35]. Distributed algorithms have much structural variety in how they perform gossip, evaluate the gradient, and update their local state variables. Furthermore, changing the order of these communication and computation steps results in different algorithms. As an example, consider the NIDS [15] algorithm update:

$$\begin{aligned} x_i^0 &\in \mathbb{R}^d \text{ arbitrary}, \quad x_i^1 = x_i^0 - \alpha \nabla f_i(x_i^0) \\ x_i^{k+2} &= \sum_{j=1}^n \widetilde{w}_{ij}\big(2x_j^{k+1} - x_j^k - \alpha \nabla f_j(x_j^{k+1}) + \alpha \nabla f_j(x_j^k)\big) \end{aligned} \tag{NIDS}$$

where $\{\widetilde{w}_{ij}\} \in \mathbb{R}^{n\times n}$ is a gossip matrix, $\alpha \in \mathbb{R}$ the stepsize, and $x_i^k \in \mathbb{R}^d$ the state of agent i at iteration k. Each update involves two previous iterates as well as a difference of gradients, similar to so-called "gradient tracking" algorithms [18]. In contrast, Exact Diffusion [46] uses two state variables, which are updated with a gradient step followed by a gossip step:

$$\begin{aligned} x_{1i}^{k+1} &= x_{2i}^k - \alpha\, \nabla f_i(x_{2i}^k) \\ x_{2i}^{k+1} &= \sum_{j=1}^n w_{ij}\, (x_{1j}^{k+1} - x_{1j}^k + x_{2j}^k). \end{aligned} \tag{Exact Diffusion}$$

While their update equations appear different, we show in Section 4.3 that NIDS and Exact Diffusion are actually equivalent, which exemplifies the need to better understand the taxonomy of distributed algorithms. To this end, we develop a canonical form that parameterizes the following broad class of first-order distributed algorithms.

Algorithm Properties. *We consider the class of distributed algorithms that satisfy the following properties.*

P1. *Each agent has a local state of dimension at most* $2d$*. Since* $\nabla f_i : \mathbb{R}^d \to \mathbb{R}$ *by assumption, this allows each agent to store up to two past iterates.*

P2. *At each iteration, each agent may do each of the following once in any order:*

(a) *communicate any number of local variables with its immediate neighbors simultaneously,*

(b) *compute its local gradient at a single point, and*

(c) *update its local state.*

P3. *The local state updates are linear, time-invariant, deterministic, and homogeneous across all agents and dimensions of the objective function.*

While NIDS and Exact Diffusion belong to the class of algorithms described by P1–P3, not all algorithms do. Exceptions include accelerated [25, 43], proximal [33], stochastic [24], and asynchronous [45] variants.

4.1 Properties

Similar to the controllable canonical form for linear time-invariant systems, we want our canonical form to have existence and uniqueness properties. That is, any algorithm satisfying P1–P3 should be equivalent to the canonical form through appropriate selection of parameters, with a one-to-one correspondence between algorithms and parameter selections. The canonical form should also be minimal in that it uses the fewest number of parameters possible to express all algorithms in the considered class.

In the canonical form, we use the graph Laplacian matrix $\mathcal{L} \in \mathbb{R}^{n\times n}$ to model local communication. This is equivalent to using a gossip matrix W with $W := I - \mathcal{L}$. We make the following assumption throughout the chapter.

Assumption 3 The Laplacian matrix $\mathcal{L} \in \mathbb{R}^{n\times n}$ is symmetric, positive semidefinite, and satisfies $\mathcal{L}\mathbf{1} = 0$, where $\mathbf{1}$ is the all-ones vector. The zero eigenvalue of $\mathcal{L}$ is distinct, implying that the underlying graph is connected.

Our canonical form satisfies properties P1–P3, and is parameterized by five real scalars: $\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3$.

Algorithm 1

Initialization: Let $\mathcal{L} \in \mathbb{R}^{n\times n}$ be a Laplacian matrix. Each agent $i \in \{1, \dots, n\}$ chooses $x_i^0 \in \mathbb{R}^d$ and $w_i^0 \in \mathbb{R}^d$.

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

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

Local communication

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

$$v_{2i}^k = \textstyle\sum_{j=1}^n \mathcal{L}_{ij}\, w_j^k \quad \text{(only required if } \zeta_2 \neq 0\text{)} \tag{C.2}$$

Local gradient computation

$$y_i^k = x_i^k - \zeta_3\, v_{1i}^k \tag{C.3}$$

$$u_i^k = \nabla f_i(y_i^k) \tag{C.4}$$

Local state update

$$x_i^{k+1} = x_i^k + \zeta_0\, w_i^k - \alpha\, u_i^k - \zeta_1 v_{1i}^k + \zeta_2 v_{2i}^k \tag{C.5}$$

$$w_i^{k+1} = w_i^k - v_{1i}^k \tag{C.6}$$

end for

end for

The algorithm requires agent i to store two local state variables, x_i^k and w_i^k. Agents first communicate their states with neighbors using the Laplacian matrix; note that w_i^k need not be transmitted if $\zeta_2 = 0$ since the result is unused in that case. Agent i then evaluates its local gradient at y_i^k, and updates its states using the results of the communication and computation. The sequence $\{y_i^k\}$ is then agent i's estimate of the optimizer $x^\star$.

Technical Conditions. *The parameters $\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3$ are constant scalars that satisfy the following:*

T1. $\alpha \neq 0$.

T2. *The linear system $\alpha u = (\zeta_0 I + \zeta_2 \mathcal{L})w$ has a solution $w \in \mathbb{R}^n$ for all $u \in \mathbb{R}^n$ that satisfies $\mathbf{1}^\mathsf{T} u = 0$.*

T3. *Either $\zeta_0 = 0$ or the algorithm is initialized such that $\sum_{i=1}^n w_i^0 = 0$. An easy way to satisfy this condition is for every agent to use the initialization $w_i^0 = 0$.*

We present the two results, which relate the distributed algorithm class to our canonical form.

Definition 4.1 (Optimal Fixed Point) The canonical form has an *optimal fixed point* if there exists a fixed point $(v_{1i}^\star, v_{2i}^\star, y_i^\star, u_i^\star, x_i^\star, w_i^\star)$ for $i \in \{1, \ldots, n\}$ such that $y_i^\star = x^\star$ for all $i \in \{1, \ldots, n\}$, where $x^\star$ is defined in (1.1).

Our first main result states that the technical conditions are *sufficient* for the canonical form to have an optimal fixed point; we prove the result in Appendix A.2.

Theorem 4.2 (Sufficiency) If technical conditions T1–T3 are satisfied, then the canonical form has a fixed point. Moreover, all fixed points are optimal fixed points.

Note that the technical conditions are sufficient for the canonical form to have an optimal fixed point, but do *not* guarantee convergence to the fixed point.

Our second main result is that the technical conditions are *necessary* for convergence to an optimal fixed point. The result also characterizes the existence and uniqueness properties of our canonical form. We prove the result in Appendix A.3.

Theorem 4.3 (Necessity) Consider a distributed algorithm that satisfies properties P1–P3 and converges to an optimal fixed point. Then there exist parameters $\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3$ for which the algorithm is equivalent to our canonical form. Furthermore, the parameters are unique, satisfy technical conditions T1–T3, and constitute a minimal parameterization of all such algorithms.

To prove the main results, we develop the notion of a transfer function for distributed algorithms in Section 4.2 and provide associated necessary conditions that hold when the algorithm converges to an optimal fixed point.

4.2 Transfer Function Interpretation

In this section, we show how to represent distributed algorithms as dynamical systems and obtain their corresponding transfer functions. We derive necessary conditions on the poles and zeros of the transfer function of any distributed algorithm that solves (1.1). These conditions are used to prove Theorem 4.3, and also give rise to an impossibility result stating that no algorithm with a single state can solve the distributed optimization problem.

The general dynamical system model for a distributed optimization algorithm satisfying P1–P3 is given by[1]

$$\begin{bmatrix} \xi_i^{k+1} \\ y_i^k \end{bmatrix} = \left(\begin{bmatrix} A_0 & B_0 \\ C_0 & D_0 \end{bmatrix} \otimes I_d \right) \begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix} + \left(\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \otimes I_d \right) \sum_{j=1}^{n} \mathcal{L}_{ij} \begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix} \tag{4.1a}$$

$$u_i^k = \nabla f_i(y_i^k) \tag{4.1b}$$

for $i \in \{1, \dots, n\}$ where $\xi_i^k \in \mathbb{R}^{2d}$ is the state, $u_i^k \in \mathbb{R}^d$ the input, and $y_i^k \in \mathbb{R}^d$ the output of agent i at iteration k. The model (4.1) over-parameterizes all algorithms in our class of interest. To simplify notation, we set $d = 1$ throughout the rest of the section, though our results hold for general $d \in \mathbb{N}$.

We can write (4.1) in vectorized form by concatenating the states of the agents as $\xi^k := \begin{bmatrix} (\xi_1^k)^\mathsf{T} & \cdots & (\xi_n^k)^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and similarly for u^k and y^k. The iterations are then

$$\xi^{k+1} = A(\mathcal{L})\xi^k + B(\mathcal{L})u^k \tag{4.2a}$$

$$y^k = C(\mathcal{L})\xi^k + D(\mathcal{L})u^k \tag{4.2b}$$

$$u^k = \nabla f(y^k) \tag{4.2c}$$

where the i^{th} component of $\nabla f(y^k)$ is $\nabla f_i(y_i^k)$, and the system matrices are:

$$A(\mathcal{L}) := I_n \otimes A_0 + \mathcal{L} \otimes A_1, \qquad B(\mathcal{L}) := I_n \otimes B_0 + \mathcal{L} \otimes B_1,$$

$$C(\mathcal{L}) := I_n \otimes C_0 + \mathcal{L} \otimes C_1, \qquad D(\mathcal{L}) := I_n \otimes D_0 + \mathcal{L} \otimes D_1.$$

By Assumption 3, we may write $\mathcal{L} = V \Lambda V^\mathsf{T}$ where $V = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix of eigenvectors and $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ is a diagonal matrix of eigenvalues with $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n$. For $\ell = 1, \dots, n$ we define the state, input, and output in direction v_ℓ as follows:

$$\bar{\xi}_\ell^k := (v_\ell^\mathsf{T} \otimes I_2)\xi^k \qquad \bar{u}_\ell^k := v_\ell^\mathsf{T} u^k \qquad \bar{y}_\ell^k := v_\ell^\mathsf{T} y^k \tag{4.3}$$

In these new coordinates, (4.2) is equivalent to the n decoupled single-input single-output systems

$$\bar{\xi}_\ell^{k+1} = (A_0 + \lambda_\ell A_1)\bar{\xi}_\ell^k + (B_0 + \lambda_\ell B_1)\bar{u}_\ell^k \tag{4.4a}$$

$$\bar{y}_\ell^k = (C_0 + \lambda_\ell C_1)\bar{\xi}_\ell^k + (D_0 + \lambda_\ell D_1)\bar{u}_\ell^k \tag{4.4b}$$

$$\bar{u}_\ell^k = v_\ell^\mathsf{T} \nabla f\big(v_1\, \bar{y}_1^k + \cdots + v_n\, \bar{y}_n^k\big) \tag{4.4c}$$

[1] The iterations (4.1) are similar to the polynomial linear protocol [9] used to study the dynamic average consensus problem.

for all $\ell \in \{1, \dots, n\}$ where $\bar{\xi}_\ell^0 \in \mathbb{R}^2$. We refer to (4.4) as the *separated system*, and the corresponding transfer functions are given by

$$G_{\lambda_\ell}(z) = (C_0 + \lambda_\ell C_1)\big(zI - (A_0 + \lambda_\ell A_1)\big)^{-1}(B_0 + \lambda_\ell B_1) + (D_0 + D_1\lambda_\ell). \tag{4.5}$$

Note that ℓ indexes the n directions in the eigenspace of L as opposed to the n agents, which are indexed by i.

Given the generic transfer function $G_{\lambda_\ell}(z)$, imposing that algorithm (4.1) has an optimal fixed point leads to properties that the transfer functions must satisfy. Here, an optimal fixed point is any point $(\xi_i^\star, y_i^\star, u_i^\star)$ such that $y_i^\star = x^\star$ solves (1.1). We summarize these in the following lemma.

Lemma 4.4 Suppose algorithm (4.1) converges to an optimal fixed point. Then the transfer function (4.5) satisfies the following properties.

1. $G_{\lambda_1}(z)$ has a pole at $z = 1$ and is marginally stable.
2. $G_{\lambda_\ell}(z)$ has a zero at $z = 1$ and is strictly stable for all $\ell = 2, \dots, n$.

Proof. Since algorithm (4.1) converges to an optimal fixed point, the separated systems (4.4) also converge to fixed points. Since $\lambda_1 = 0$ and $v_1 = \frac{1}{\sqrt{n}}1$, and by (4.3) and (1.1),

$$\bar{u}_1^\star = v_1^\mathsf{T} u^\star = \tfrac{1}{\sqrt{n}} \sum_{i=1}^n \nabla f_i(y_i^\star) = \sqrt{n}\, \nabla f(x^\star) = 0. \tag{4.6a}$$

For $\ell = 2, \dots, n$, each v_ℓ is orthogonal to v_1 since L is symmetric. Again using (4.3) and (1.1),

$$\bar{y}_\ell^\star = v_\ell^\mathsf{T} y^\star = v_\ell^\mathsf{T} 1 x^\star = 0 \qquad \text{for } \ell \in \{2, \dots, n\}. \tag{4.6b}$$

We first prove that $G_{\lambda_1}(z)$ has a pole at $z = 1$ and is marginally stable. Suppose by contradiction that all poles of $G_{\lambda_1}(z)$ are strictly stable. Then for every input sequence, $\bar{u}_1^k \to \bar{u}_1^\star = 0$ by (4.6a), and the corresponding output sequence $\bar{y}_1^k$ must tend to $\bar{y}_1^\star = 0$. However, this is a contradiction because $\bar{y}_1^\star$ is nonzero in general. Alternatively suppose that $G_{\lambda_1}(z)$ has an unstable pole. Then there exists an input sequence such that $\bar{y}_1^k \to \infty$ as $\bar{u}_1^k \to u^\star$. This contradicts that $\bar{y}_1^\star$ is finite. Hence, $G_{\lambda_1}(z)$ is marginally stable with a pole at $z = 1$.

To prove that $G_{\lambda_\ell}(z)$ is strictly stable for $\ell = 2, \dots, n$, assume the contrary. Then there exists a pole for some z such that $|z| \geq 1$. Consequently, for a converging input sequence, $\bar{y}_\ell^k \to \bar{y}_\ell^\star \neq 0$, which contradicts (4.6b).

Finally, we show that $G_{\lambda_\ell}(z)$ has a zero at $z = 1$. Let $\bar{u}_\ell^k \to \bar{u}_\ell^\star$, which in general is a nonzero constant. Then the steady-state gain to y_ℓ must be zero to ensure (4.6b) holds. ■

The requirements on the transfer function in Lemma 4.4 imply that no single-state algorithm of the form (4.1) can solve the distributed optimization problem. This provides an explanation as to why the Distributed Gradient Descent algorithm must use a diminishing stepsize [19]. We therefore restrict ourselves to algorithms with two states (i.e., $\xi_i^k \in \mathbb{R}^{2d}$) since these are the simplest algorithms that can achieve linear convergence rates.

Corollary 4.5 No algorithm satisfying properties P2 and P3 where each agent has a local state of dimension d can solve the distributed optimization problem (1.1).

Proof. Such an algorithm can be written in the form (4.1) with $A_i, B_i, C_i, D_i \in \mathbb{R}$ for $i = 1, 2$ (since the state is dimension d) and $D_0 = D_1 = 0$. The corresponding transfer function then has the form

$$G_{\lambda_\ell}(z) = \frac{(C_0 + \lambda_\ell C_1)(B_0 + \lambda_\ell B_1)}{z - (A_0 + \lambda_\ell A_1)}.$$

Suppose the algorithm solves the distributed optimization problem (1.1). Then from Lemma 4.4, $G_{\lambda_0}(z)$ must have a pole at $z = 1$, and $G_{\lambda_\ell}(z)$ must have a zero at $z = 1$ for all $\ell = 2, \dots, n$. The first condition implies that $A_0 = 1$ with $B_0 C_0 \neq 0$, while the second condition implies that either $B_0 = B_1 = 0$ or $C_0 = C_1 = 0$. These conditions contradict each other, implying that the algorithm does *not* solve the distributed optimization problem. ■

4.3 Examples

We now outline a procedure to put any distributed optimization algorithm in our class of interest in canonical form. Note that the canonical form has the form (4.1) with

$$\left[\begin{array}{c|c} A_0 & B_0 \\ \hline C_0 & D_0 \end{array}\right] = \left[\begin{array}{cc|c} 1 & \zeta_0 & -\alpha \\ 0 & 1 & 0 \\ \hline 1 & 0 & 0 \end{array}\right] \quad \text{and} \quad \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] = \left[\begin{array}{cc|c} -\zeta_1 & \zeta_2 & 0 \\ -1 & 0 & 0 \\ \hline -\zeta_3 & 0 & 0 \end{array}\right] \tag{4.7}$$

and the corresponding transfer function (4.5) is

$$G^{\text{CF}}_{\lambda_\ell}(z) = -\frac{\alpha\,(1 - \zeta_3\lambda_\ell)(z-1)}{(z-1)(z - 1 + \zeta_1\lambda_\ell) + \lambda_\ell\,(\zeta_0 + \zeta_2\lambda_\ell)}. \tag{4.8}$$

We can then apply the following procedure to put any distributed algorithm satisfying properties P1–P3 into our canonical form. We summarize the results for several known algorithms in Table 4.1.

1. Write the algorithm in the form (4.1) to obtain the system matrices (A_0, B_0, C_0, D_0) and (A_1, B_1, C_1, D_1).
2. Compute the transfer function $G_{\lambda_\ell}(z)$ using (4.5).
3. Obtain parameters $\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3$ by comparing coefficients of the transfer function $G_{\lambda_\ell}(z)$ with that of the canonical form $G^{\text{CF}}_{\lambda_\ell}(z)$ in (4.8).

Since the parameters of our canonical form are unique, we can compare various algorithms by putting them in this form. For example, from Table 4.1 we observe that NIDS and Exact Diffusion have the same parameters and are therefore equivalent. Similarly, for the algorithm of Jakovetić, choosing $\mathcal{B} = 0$ recovers DIGing, and $\mathcal{B} = \frac{1}{\alpha}W$ with $W \mapsto \frac{1}{2}(I + W)$ recovers EXTRA, as noted in [10].

[2]The similar algorithm AugDGM [44] does not fit in our framework because it requires two sequential rounds of communication per iteration.

Table 4.1 Parameters of distributed optimization algorithms in canonical form using $W = I - \mathcal{L}$.

	ζ_0	ζ_1	ζ_2	ζ_3
EXTRA [32]	$\frac{1}{2}$	1	0	0
NIDS [15]	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Exact Diffusion [46]	$\frac{1}{2}$	1	0	$\frac{1}{2}$
DIGing [18, 26]	0	2	1	0
AsynDGM^2 [45]	0	2	1	1
Jakovetić [10] ($\mathcal{B} = \beta I$)	$\alpha\beta$	2	1	0
Jakovetić [10] ($\mathcal{B} = \beta W$)	$\alpha\beta$	2	$1 - \alpha\beta$	0

Remark 4.6 To provide an additional degree of freedom, we can relate the gossip matrix to the Laplacian matrix by $W = I - \mu\mathcal{L}$ where $\mu \in \mathbb{R}$ is an additional scalar parameter. This is equivalent to choosing the gossip matrix as a convex combination of $I - \mathcal{L}$ and the identity, i.e., $W = (1 - \mu)I + \mu(I - \mathcal{L})$.

Remark 4.7 (Uniqueness of Realization) The canonical form (4.7) can be reliably obtained from an algorithm by computing its transfer function (4.8) and then extracting coefficients. In contrast, working with realizations directly can be problematic because even if coordinates are chosen such that (A_0, B_0, C_0, D_0) has the desired form, this does not ensure uniqueness of (A_1, B_1, C_1, D_1). For example, one can easily check that the following realization has the same transfer function as (4.7):

$$\left[\begin{array}{c|c} A_0 & B_0 \\ \hline C_0 & D_0 \end{array}\right] = \left[\begin{array}{cc|c} 1 & \zeta_0 & -\alpha \\ 0 & 1 & 0 \\ \hline 1 & 0 & 0 \end{array}\right] \quad \text{and} \quad \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] = \left[\begin{array}{cc|c} -\zeta_1 & \zeta_2 & \alpha\zeta_3 \\ -1 & 0 & 0 \\ \hline 0 & 0 & 0 \end{array}\right].$$

The equivalence class of realizations for a doubly-indexed transfer function (i.e. with z and λ) is more involved than the singly-indexed case and a broader discussion on this topic is outside the scope of the present work; seminal references include [8, 29].

NIDS. To illustrate our approach, we return to the NIDS [15] algorithm. As suggested by the authors of [15], we choose $\widetilde{W} := (I + W)/2$ where the gossip matrix W is related to the Laplacian matrix $\mathcal{L}$ by $W = I - \mathcal{L}$.

First, we write NIDS in the form (4.1) as follows:

$$\begin{bmatrix} x^{k+2} \\ x^{k+1} \\ \nabla f(y^k) \end{bmatrix} = \begin{bmatrix} 2I - L & \frac{1}{2}\mathcal{L} - I & \alpha(I - \frac{1}{2}\mathcal{L}) \\ I & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{k+1} \\ x^k \\ \nabla f(y^{k-1}) \end{bmatrix} + \begin{bmatrix} \alpha(\frac{1}{2}\mathcal{L} - I) \\ 0 \\ I \end{bmatrix} \nabla f(y^k)$$

$$y^k = \begin{bmatrix} I & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{k+1} \\ x^k \\ \nabla f(y^{k-1}) \end{bmatrix}$$

This is equivalent to (4.1) with state-space matrices

$$\left[\begin{array}{c|c} A_0 & B_0 \\ \hline C_0 & D_0 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & -1 & \alpha & -\alpha \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \end{array}\right] \quad \text{and} \quad \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] = \left[\begin{array}{ccc|c} -1 & \frac{1}{2} & -\frac{\alpha}{2} & \frac{\alpha}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right],$$

which have associated transfer function (4.5) given by

$$G_{\lambda_\ell}(z) = -\frac{\alpha\,(1 - \frac{1}{2}\lambda_\ell)(z-1)}{(z-1)(z-1+\lambda_\ell) + \frac{1}{2}\lambda_\ell}.$$

Comparing coefficients with that of $G^{\text{CF}}_{\lambda_\ell}(z)$ in (4.8), we find that the parameters of the canonical form are $(\zeta_0, \zeta_1, \zeta_2, \zeta_3) = (\frac{1}{2}, 1, 0, \frac{1}{2})$.

Chapter 5

Algorithm Design

We now turn to the problem of algorithm *synthesis*, or design, as discussed in [36]. Given a class of allowable functions and graphs, we seek to find the plant (algorithm) that achieves the best (fastest) rate of convergence. The topic of this chapter complements our prior study of algorithm analysis, wherein the plant is known and we want to certify the rate of convergence. Now, we will use the SDP (3.10) to design a distributed optimization algorithm, which we name SVL. Our guiding principle is to seek the fastest possible rate bound guarantee while keeping the algorithm as simple as possible. Therefore, we seek an algorithm with two states that only requires one state to be communicated at every timestep. Motivated by our canonical form for distributed algorithms over time-invariant graphs [35], we restrict our search to algorithms parameterized as

$$\left[\begin{array}{c|c|c} A & B_u & B_v \\ \hline C_y & D_{yu} & D_{yv} \\ \hline C_z & D_{zu} & D_{zv} \\ \hline F_x & F_u & \end{array}\right] = \left[\begin{array}{cc|c|c} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \\ \hline 1 & 0 & 0 & -\delta \\ \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & \end{array}\right]. \tag{5.1}$$

As long as $\beta \neq 0$, this algorithm satisfies the fixed point conditions of Proposition 3.5. Moreover, the update equations satisfy the conditions for implementability (3.2) and therefore do not contain circular dependencies, so we can implement the algorithm in a straightforward fashion as in Algorithm 2. 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. To begin, we show how to use the SDP (3.10) to choose the algorithm parameters.

5.1 Choosing the Algorithm Parameters

The problem of minimizing the worst-case convergence rate ρ over the algorithm parameters $(\alpha, \beta, \gamma, \delta)$ and SDP solution (P, Q, R), subject to the SDP being feasible, is difficult due to the nonlinear matrix inequalities. Instead, we show that for a particular choice of (α, γ, δ), the remaining parameters (β, ρ) can be chosen such that the SDP is

Algorithm 2 (template for the SVL algorithm)

Initialization: Let $\mathcal{L}^k \in \mathbb{R}^{n\times n}$ be a Laplacian matrix. Agents $i \in \{1,\dots,n\}$ choose initial local state $x_i^0 \in \mathbb{R}^d$ arbitrarily and $w_i^0 \in \mathbb{R}^d$ such that $\sum_{i=1}^n w_i^0 = 0$ (e.g. $w_i^0 = 0$).

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

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

Local communication

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

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

feasible, where the matrix in the disagreement LMI (3.10b) is rank one. Extensive numerical optimizations of the SDP suggest that the optimal parameters do in fact have this structure.

We now state our main design result, which describes the convergence rate of the SVL algorithm. A proof is provided in Appendix A.4.

Theorem 5.1 (SVL) Consider applying Algorithm 2 to the distributed optimization problem (1.1), and suppose Assumptions 1 and 2 hold with $0 < m < L$ and $0 \le \sigma < 1$. Define $\eta := 1 + \rho - \kappa\,(1-\rho)$ and choose the parameters

$$\alpha = \frac{1-\rho}{m}, \qquad \gamma = 1 + \beta, \qquad \delta = 1, \tag{5.2}$$

where β and $\rho \in \left[\frac{L-m}{L+m}, 1\right)$ satisfy the constraints

$$\big(2\beta - (1-\rho)(\kappa+1)\big)(\beta - 1 + \rho^2) < 0, \tag{5.3a}$$

$$\rho^2 \left(\frac{\beta - 1 + \rho^2}{\beta - 1 + \rho}\right)\left(\frac{2 - \eta - 2\beta}{2\rho^2\beta - (1-\rho^2)\eta}\right)\left(\frac{(2\rho^2+\eta)\beta - (1-\rho^2)\eta}{(1+\rho)(\eta - 2\eta\rho + 2\rho^2) - (2\rho^2+\eta)\beta}\right) = \sigma^2. \tag{5.3b}$$

Then there exists a constant $c > 0$ independent of i and k such that for all agents $i \in \{1,\dots,n\}$ and all iterations $k \ge 0$, $\|y_i^k - y_{\text{opt}}\| \le c\,\rho^k$ where $y_{\text{opt}} \in \mathbb{R}^d$ is the optimizer of (1.1).

Theorem 5.1 provides conditions on parameters $(\alpha, \beta, \gamma, \delta)$ of Algorithm 2 such that the algorithm converges with rate no slower than 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 (5.3). 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 (5.3b), but this equation does not have a closed-form solution for ρ. Instead, we consider fixing the rate ρ and maximizing the corresponding spectral gap. We can then choose β to maximize σ^2 in (5.3b). We do this for computational ease; maximizing σ for a given ρ is akin to looking for the sparsest graph for which the given rate holds. Maximizing σ works because the convergence rate only worsens as the graph becomes disconnected. In other words, ρ is a nonstrictly increasing function of σ. We find the value of β that maximizes σ^2 for a fixed convergence rate ρ by setting the derivative equal to zero:

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

where the coefficients s_i are given by

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

Solving the first factor for β, we find that it does not satisfy the inequality (5.3a) 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

$$s_0 + s_1\beta + s_2\beta^2 + s_3\beta^3 = 0 \tag{5.4}$$

to find the unique real solution β that satisfies (5.3a). Substituting this value of β into (5.3b) gives the solution for σ. Denote the solution by $\hat{\sigma}$. If $\hat{\sigma} < \sigma$, we increase ρ; otherwise, we decrease ρ. We then repeat until $\hat{\sigma}$ is sufficiently close to σ. Algorithm 3 summarizes this procedure for finding the parameters β and ρ that optimize the worst-case convergence rate. We define SVL to be Algorithm 2 with parameters chosen using Algorithm 3.

Fig. 5.1 displays the worst-case convergence rate ρ as a function of the spectral gap σ and the centralized gradient rate $\frac{\kappa-1}{\kappa+1}$ for SVL. 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 by the problem of having agents agree on the solution (i.e., consensus). This corresponds to the horizontal lines for small values of σ in the top panel of Fig. 5.1. 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 bottom panel of Fig. 5.1.

Algorithm 3 (computing the SVL parameters)

Initialization: Let $0 < m < L$, $0 \leq \sigma < 1$, and $\epsilon > 0$. Define $\kappa := L/m$. Set $\rho_1 = 0$ and $\rho_2 = 1$.

while $\rho_2 - \rho_1 > \epsilon$ **do**

$\rho = (\rho_1 + \rho_2)/2$

Let β be the unique real solution to (5.4) that satisfies (5.3a).

Using this value of β, let $\hat{\sigma}$ denote the solution to (5.3b).

if $\hat{\sigma} < \sigma$ **then**

$\rho_1 = \rho$

else

$\rho_2 = \rho$

end if

end while

return ρ, β

Remark 5.2 (Optimality) We conjecture that the SVL parameters $(\alpha, \beta, \gamma, \delta)$ produce the fastest worst-case convergence rate over all algorithms in the form of Algorithm 2, that is certifiable using the main analysis SDP. However, we make no formal claims of optimality of the SVL algorithm.

5.2 Interpretation of SVL as Inexact ADMM

To motivate the structure of SVL, we show how SVL can be interpreted as an inexact version of the alternating direction method of multipliers (ADMM). Using the formulation in [4, Section 7.1], the distributed optimization problem (1.1) can be solved using ADMM:

$$x_i^{k+1} = \arg\min_x f_i(x) + (x - y_i^k)^\mathsf{T} z_i^k + \tfrac{\beta}{2}\|x - y_i^k\|^2 \tag{5.5a}$$

$$y_i^{k+1} = \frac{1}{n}\sum_{j=1}^n x_j^{k+1} \tag{5.5b}$$

$$z_i^{k+1} = z_i^k + \beta\,(x_i^{k+1} - y_i^{k+1}) \tag{5.5c}$$

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 (5.5a) *exactly* as well as compute the *exact* average (5.5b) at each iteration. Instead, we consider a variant where the computations and communications are *inexact*. Specifically, we replace the exact minimization (5.5a) with a single gradient step with initial condition y_i^k and stepsize $\alpha > 0$, and we replace the exact averaging step (5.5b) 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 2 with $\gamma = 1+\beta$ and $\delta = 1$. In other words, SVL corresponds to an inexact version of ADMM, where α is the stepsize of the gradient step and β is the ADMM parameter. Other distributed ADMM variants are found in [5, 31].

5.3 Special Cases

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

Single Agent. When $n = 1$, the distributed optimization problem is equivalent to centralized optimization. In this case, the Laplacian matrix is simply the scalar $\mathcal{L}^k = 0$, so $v_1^k = 0$ for all $k \geq 0$. Algorithm 2 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 of $\rho = \frac{\kappa-1}{\kappa+1}$ is achieved when $\alpha = \frac{2}{L+m}$.

Unity Condition Ratio. When the condition ratio $\kappa = 1$ (i.e., $m = L$), the distributed optimization problem 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 $f_i^k(x) = \frac{L}{2}\|x - r_i^k\|^2$, where $r_i^k \in \mathbb{R}^d$ is a parameter on agent $i \in \{1, \ldots, n\}$ at iteration k. The SVL algorithm then 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 [12]. When the objective functions are constant, the r_i terms cancel from the iterations and only affect the initial conditions. This case is referred to as *static* average consensus [37], and the worst-case rate of convergence is $\rho = \sigma$ [39].

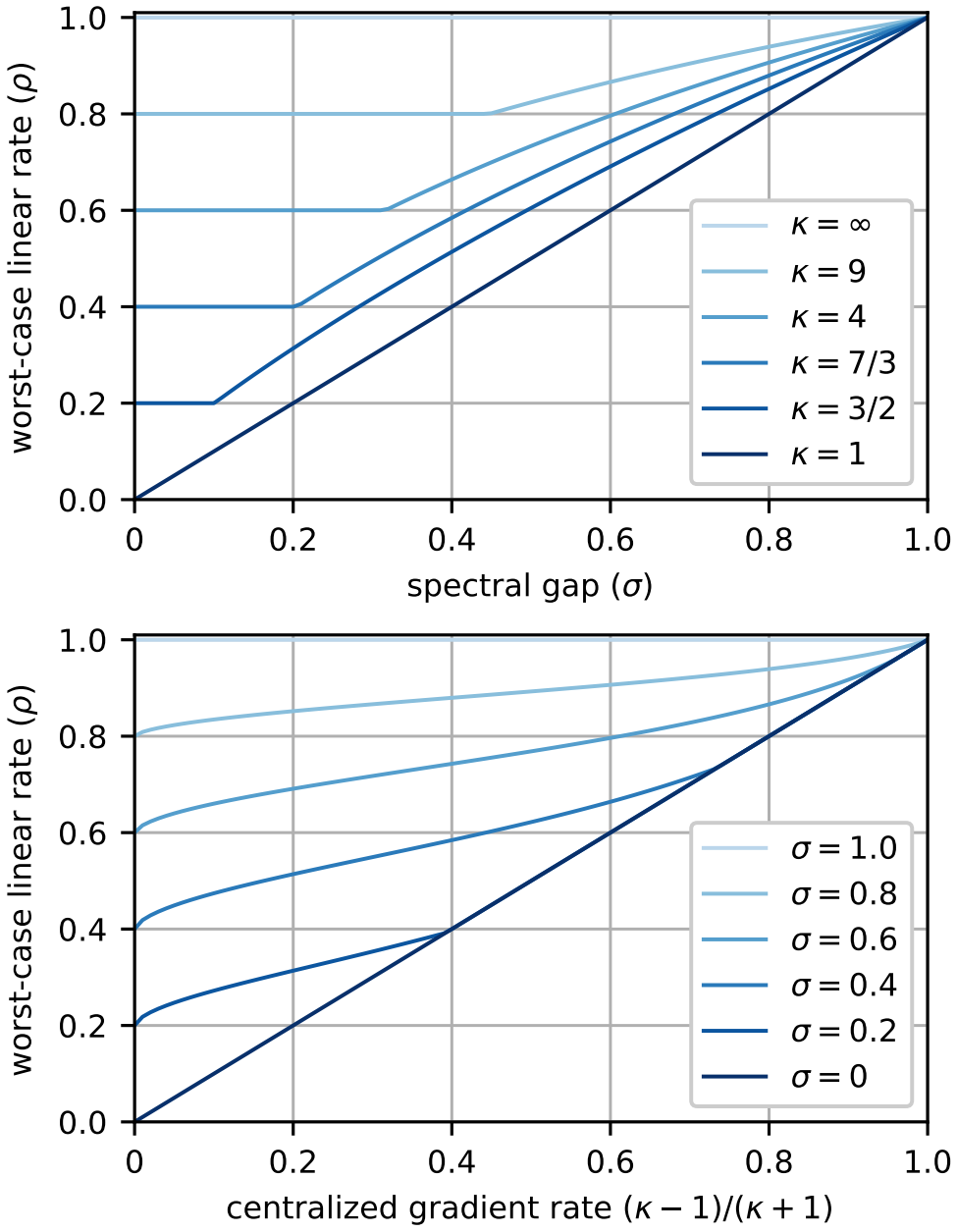

Figure 5.1 Worst-case linear rate ρ of SVL in Theorem 5.1 as a function of κ and σ. Top plot: as $\kappa \to 1$ (quadratic objective), we obtain $\rho = \sigma$ (optimal linear consensus rate). Bottom plot: as $\sigma \to 0$ (fully connected graph), we obtain $\rho = \frac{\kappa-1}{\kappa+1}$ (optimal centralized gradient rate).

Chapter 6

Finding Lower Bounds

In an effort to show that the upper bounds for each algorithm in Fig 3.2 were likely tight, we searched for signals $\{x^k, u^k, v^k, y^k, z^k\}$ that satisfied (3.1) for some choice of f_i and $\mathcal{L}^k$ satsifying Assumptions 1 and 2, respectively [36].

We first solved a relaxed version of the problem, where we replaced Assumptions 1 and 2 by the weaker conditions (3.13) and (3.14), respectively. We used the following greedy heuristic. For a given algorithm and rate ρ, we solved (3.10) to obtain (P, Q, R). At each time step k, we then maximized the Lyapunov increment $V^{k+1} - \rho^2 V^k$, where V^k is defined in (3.12). We solved the following optimization problem for $k \geq 0$.

$$\begin{array}{ll} \underset{u_i^k, v_i^k \in \mathbb{R}^d}{\max} & V^{k+1} - \rho^2 V^k \\ \text{such that} & \text{(3.1a), (3.1c), and (3.14) hold,} \\ & \text{(3.13) holds for } i = 1, \ldots, n, \\ & 1^\mathsf{T} v^k = 0. \end{array} \tag{6.1}$$

For $k = 0$, we also included x^0 as an optimization variable and the normalization $V^0 = 1$. For $k \geq 1$, we solved (6.1) using the x^k found at the previous iteration and warm-starting u^k, v^k. We used the Ipopt [38] local solver with default settings since (6.1) is a nonconvex quadratically constrained quadratic program. Note that we must choose parameters n and d.

Our relaxed heuristic using $n = d = 2$ was successful in constructing trajectories that matched the worst-case bounds from (3.10). To illustrate, we simulated EXTRA, NIDS, DIGing, and SVL with $\kappa = 10$ and a few values of σ in Fig. 6.1. For each trajectory, we plotted $\|y^k - y^\star\|$ together with the corresponding upper bound ρ found from Theorem 3.10. We obtained similar results for the other algorithms from Table 3.1.

Since we used the relaxation (3.14) to construct z^k and v^k, there is no guarantee that there will exist a *linear* Laplacian $\mathcal{L}^k$ such that $v^k = \mathcal{L}^k z^k$. However, finding whether such an $\mathcal{L}^k$ exists amounts to solving a convex

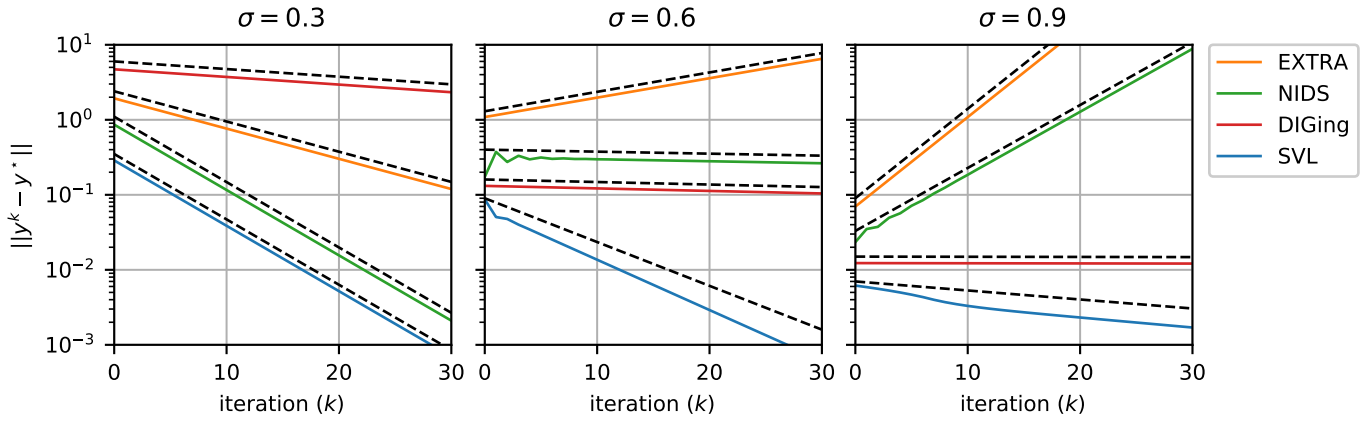

Figure 6.1 Approximate worst-case trajectories for EXTRA, NIDS, DIGing, and SVL. Trajectories were found by solving the relaxed problem (6.1). We used α optimized as in Fig. 3.2 and the default $\mu = 1$. Simulations were performed for $\kappa = 10$, $\sigma \in \{0.3, 0.6, 0.9\}$, and $n = d = 2$. Dashed lines indicate corresponding upper bounds obtained from Theorem 3.10 and shown in Fig. 3.2. All traces were vertically translated to improve clarity.

optimization problem:

$$\begin{aligned} \min_{\mathcal{L}^k \in \mathbb{R}^{n\times n}} \quad & \|I - \Pi - \mathcal{L}^k\| \\ \text{such that} \quad & (\mathcal{L}^k \otimes I) z^k = v^k, \\ & \mathcal{L}^k 1 = 0, \quad 1^\mathsf{T} \mathcal{L}^k = 0. \end{aligned} \tag{6.2}$$

If (6.2) is feasible and its optimal value is less than or equal to σ, then the associated L^k is a valid Laplacian matrix at timestep k. While there is no guarantee that (6.2) will even be feasible, we reasoned that since there are n^2 variables and $2n + ndc$ linear constraints, where d and c are the number of rows of C_y and C_z, respectively, we could increase our chances of finding feasible $\mathcal{L}^k$ with n large and d and c small.

In Figure 6.2, we show a successful construction for the NIDS algorithm, which has $c = 1$. We solved (6.1) with $n = 15$ and $d = 1$, and solved (6.2) at each timestep. An optimal cost for (6.2) of σ was always achieved.

This result indicates that the upper bound for NIDS in Fig. 3.2 is likely tight, and that NIDS is not robustly stable in the time-varying setting. In other words, the *network-independent* rate bound enjoyed by NIDS in the constant-graph setting [15, Thm. 2] does not carry over to the time-varying setting.

Remark 6.1 There may be other approaches to finding a worst-case $\mathcal{L}^k$ that outperform the outlined approach using (6.2). For example, one might try alternating convex optimizations or including $\mathcal{L}^k$ directly as an optimization variable in a nonlinear program.

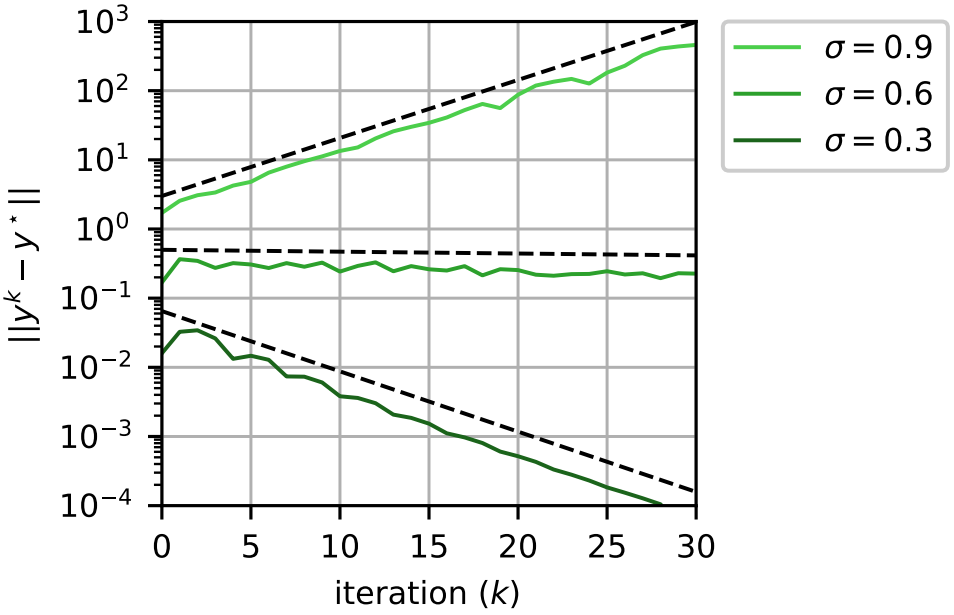

Figure 6.2 Worst-case trajectories for NIDS found by solving (6.1) and successfully solving (6.2) to construct a sequence of Laplacians $\{\mathcal{L}^k\}$. Simulations were performed using optimized α, $\mu = 1$, $\kappa = 10$, $n = 15$, and $d = 1$ for $\sigma \in \{0.3, 0.6, 0.9\}$. Trajectories were plotted with their accompanying rate bounds (dashed lines) from Theorem 3.10 and translated to improve clarity.

LIST OF REFERENCES

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

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

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

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, volume 3. Foundations and Trends in Machine Learning, 2010.

[5] T. Chang, M. Hong, and X. Wang. Multi-agent distributed optimization via inexact consensus admm. *IEEE Transactions on Signal Processing*, 63(2):482–497, 2015.

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

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

[8] E. Fornasini and G. Marchesini. Doubly-indexed dynamical systems: State-space models and structural properties. *Theory of Computing Systems*, 12(1):59–72, 1978.

[9] R.A. Freeman, T.R. Nelson, and K.M. Lynch. A complete characterization of a class of robust linear average consensus protocols. In *Proc. of the 2010 Amer. Control Conf.*, pages 3198–3203, 2010.

[10] Dušan Jakovetić. A unification and generalization of exact distributed first-order methods. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):31–46, 2018.

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

[12] Solmaz S. Kia, Bryan Van Scoy, Jorge Cortés, Randy A. Freeman, Kevin M. Lynch, and Sonia Martínez. Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. *IEEE Control Systems Magazine*, 39(3):40–72, 2019.

[13] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[14] Laurent Lessard and Peter Seiler. Direct synthesis of iterative algorithms with bounds on achievable worst-case convergence rate. *American Control Conference*, 2020.

[15] Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019.

[16] Qing Ling and Zhi Tian. Decentralized sparse signal recovery for compressive sleeping wireless sensor networks. *IEEE Transactions on Signal Processing*, 58(7):3816–3827, 2010.

[17] Patrick Kofod Mogensen and Asbjørn Nilsen Riseth. Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24), 2018.

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

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

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

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

[22] B. T. Polyak. *Introduction to optimization*. Optimization Software, Publications Division, New York, 1987.

[23] Joel B. Predd, Sanjeev R. Kulkarni, and H. Vincent Poor. A collaborative training algorithm for distributed learning. *IEEE Transactions on Information Theory*, 55(4):1856–1871, 2009.

[24] Shi Pu and Angelia Nedić. A distributed stochastic gradient tracking method. In *IEEE Conference on Decision and Control*, pages 963–968, 2018.

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

[26] Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 2017.

[27] Michael Rabbat and Robert Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 20–27. ACM, 2004.

[28] Sundhar Srinivasan Ram, Venugopal V. Veeravalli, and Angelia Nedić. Distributed non-autonomous power control through distributed convex optimization. In *IEEE INFOCOM*, pages 3001–3005, 2009.

[29] R. Roesser. A discrete state-space model for linear image processing. *IEEE Transactions on Automatic Control*, 20(1):1–10, 1975.

[30] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent 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.

[31] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

[32] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[33] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63(22):6013–6023, 2015.

[34] Akhil Sundararajan, Bin Hu, and Laurent Lessard. Robust convergence analysis of distributed optimization algorithms. In *Allerton Conference on Communication, Control, and Computing*, pages 1206–1212, 2017.

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

[36] Akhil Sundararajan, Bryan Van Scoy, and Laurent 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, 2020.

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

[38] Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

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

[40] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing*, 67(1):33–46, 2007.

[41] Ran Xin, Dusan Jakovetić, and Usman A. Khan. Distributed nesterov gradient methods over arbitrary graphs. *IEEE Signal Processing Letters*, 2019.

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

[43] Ran Xin and Usman A Khan. Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking. *IEEE Transactions on Automatic Control*, 2019.

[44] Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *IEEE Conference on Decision and Control*, pages 2055–2060, 2015.

[45] Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. Convergence of asynchronous distributed gradient methods over stochastic networks. *IEEE Transactions on Automatic Control*, 63(2):434–448, 2018.

[46] Kun Yuan, Bicheng Ying, Xiaochuan Zhao, and Ali H Sayed. Exact diffusion for distributed optimization and learning—Part I: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3):708–723, 2018.

[47] Kun Yuan, Bicheng Ying, Xiaochuan Zhao, and Ali H Sayed. Exact diffusion for distributed optimization and learning—Part II: Convergence analysis. *IEEE Transactions on Signal Processing*, 67(3):724–739, 2018.

APPENDIX

A.1 Proof of Proposition 3.13

From the definition of the matrix norm and Assumption 2, we have that

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

Without loss of generality, $y = \Pi\eta + (I - \Pi)\,\phi$, where η and ϕ are arbitrary. By orthogonality, $\|y\|^2 = \|\Pi\eta\|^2 + \|(I - \Pi)\,\phi\|^2$. Substituting the decomposition of y into the above inequality,

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

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

$$\begin{bmatrix}\phi \\ \mathcal{L}^k\phi\end{bmatrix}^{\mathsf{T}} \left(M_1 \otimes (I - \Pi)\right) \begin{bmatrix}\phi \\ \mathcal{L}^k\phi\end{bmatrix} \geq 0 \tag{A.1}$$

for all $\phi \in \mathbb{R}^n$. Now let p denote the dimension of z_i^k. Then since $R \succeq 0$, it has the decomposition

$$R = \sum_{\ell=1}^{p} \mu_\ell\, w_\ell w_\ell^{\mathsf{T}},$$

where $w_\ell \in \mathbb{R}^p$ and $\mu_\ell \geq 0$. Then using that $\tilde{v}^k = (\mathcal{L}^k \otimes I_p)\, \tilde{z}^k$, the quadratic form is

$$\begin{aligned}
&\begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix}^{\mathsf{T}} \big(M_1 \otimes (I - \Pi) \otimes R\big) \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix} \\
&\qquad = \sum_\ell \mu_\ell \begin{bmatrix} \star \end{bmatrix}^{\mathsf{T}} \big(M_1 \otimes (I - \Pi)\big) \begin{bmatrix} (I \otimes w_\ell^{\mathsf{T}})\, \tilde{z}^k \\ (I \otimes w_\ell^{\mathsf{T}})\, \tilde{v}^k \end{bmatrix} \\
&\qquad = \sum_\ell \mu_\ell \begin{bmatrix} \star \end{bmatrix}^{\mathsf{T}} \big(M_1 \otimes (I - \Pi)\big) \begin{bmatrix} (I \otimes w_\ell^{\mathsf{T}})\, \tilde{z}^k \\ \mathcal{L}^k\, (I \otimes w_\ell^{\mathsf{T}})\, \tilde{z}^k \end{bmatrix},
\end{aligned}$$

which is nonnegative from (A.1) with $\phi \leftarrow (I \otimes w_\ell^{\mathsf{T}})\, \tilde{z}^k$.

A.2 Proof of Theorem 4.2

First, we show that all fixed points are optimal. Consider a fixed point of the algorithm, given by $(v_{1i}^\star, v_{2i}^\star, y_i^\star, u_i^\star, x_i^\star, w_i^\star)$ for $i \in \{1, \ldots, n\}$. Applying (C.6) and (C.1) to the fixed point, we have $0 = v_{1i}^\star = \sum_{j=1}^n L_{ij}\, x_j^\star$. This implies $y_i^\star = x_i^\star$, and $x_i^\star = x_j^\star$ for all $i, j \in \{1, \ldots, n\}$ since the graph is connected by Assumption 3. This shows that the agents reach consensus at the fixed point; we have left to show that the consensus variable is the solution to (1.1). To do so, we sum both sides of the x_i update in (C.5) to obtain

$$\alpha \sum_{i=1}^n u_i^\star = \zeta_0 \sum_{i=1}^n w_i^\star - \zeta_1 \sum_{i,j=1}^n \mathcal{L}_{ij} x_j^\star + \zeta_2 \sum_{i,j=1}^n \mathcal{L}_{ij} w_j^\star. \tag{A.2}$$

The last two terms on the right side are both zero since the Laplacian matrix is symmetric and therefore satisfies $\mathcal{L}^\mathsf{T} 1 = 0$. Furthermore, the first term on the right side is zero due to T3. To see this, we sum (C.6) to obtain $\sum_{i=1}^n w_i^{k+1} = \sum_{i=1}^n w_i^k$, so either $\zeta_0 = 0$ or $\sum_{i=1}^n w_i^\star = 0$. Finally, $\alpha \neq 0$ from T1, so we must have $0 = \sum_{i=1}^n u_i^\star = \sum_{i=1}^n \nabla f_i(y_i^\star)$. The fixed point satisfies (1.1) and is therefore optimal.

Next, we turn to showing existence of an optimal fixed point. Define y_{opt} as in (1.1). We show how to construct an optimal fixed point of the canonical form. Define $v_{1i}^\star := 0$, $y_i^\star := y_{opt}$, $u_i^\star := \nabla f_i(y_{opt})$, and $x_i^\star := y_{opt}$ for all $i \in \{1, \ldots, n\}$. We have $\sum_{i=1}^n u_i^\star = 0$ from the definition of y_{opt}, so from T2, the linear system

$$\alpha u_i^\star = \zeta_0 w_i^\star + \zeta_2 \sum_{j=1}^n \mathcal{L}_{ij}\, w_j^\star \tag{A.3}$$

has a solution $w_i^\star$. Finally, we define $v_{2i}^\star := \sum_{j=1}^n \mathcal{L}_{ij}\, w_i^\star$. Then the point $(v_{1i}^\star, v_{2i}^\star, y_i^\star, u_i^\star, x_i^\star, w_i^\star)$ for $i \in \{1, \ldots, n\}$ is an optimal fixed point.

A.3 Proof of Theorem 4.3

Consider an algorithm that satisfies properties P1–P3 and converges to an optimal fixed point. Such an algorithm can be represented by the iterations (4.1) with the following: i) $A_i \in \mathbb{R}^{2\times 2}$ for $i = 1, 2$ (since the local state vector is in $\mathbb{R}^{2d}$, ii) $D_0 = D_1 = 0$, and iii) either $B_1 = 0$ or $C_1 = 0$ (otherwise the algorithm would require two sequential rounds of communication per iteration). We can therefore parameterize the state-space matrices as

$$\left[\begin{array}{c|c} A_0 & B_0 \\ \hline C_0 & D_0 \end{array}\right] = \left[\begin{array}{cc|c} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ \hline c_1 & c_2 & 0 \end{array}\right], \qquad \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] = \left[\begin{array}{cc|c} a_5 & a_6 & b_3 \\ a_7 & a_8 & b_4 \\ \hline c_3 & c_4 & 0 \end{array}\right].$$

The corresponding transfer function has the form

$$G_\lambda(z) = \frac{(\eta_1 + \eta_2\lambda + \eta_3\lambda^2)z + (\eta_4 + \eta_5\lambda + \eta_6\lambda^2 + \eta_7\lambda^3)}{z^2 + (\eta_8 + \eta_9\lambda)z + (\eta_{10} + \eta_{11}\lambda + \eta_{12}\lambda^2)}$$

where the parameters $\{\eta_i\}$ are defined as follows:

$$\begin{aligned}
\eta_1 &:= b_1c_1 + b_2c_2 \\
\eta_2 &:= b_1c_3 + b_3c_1 + b_2c_4 + b_4c_2 \\
\eta_3 &:= b_3c_3 + b_4c_4 \\
\eta_4 &:= -a_1b_2c_2 + a_2b_2c_1 + a_3b_1c_2 - a_4b_1c_1 \\
\eta_5 &:= a_2b_2c_3 - a_1b_4c_2 - a_1b_2c_4 + a_2b_4c_1 \\
&\quad + a_3b_1c_4 + a_3b_3c_2 - a_4b_1c_3 - a_4b_3c_1 \\
&\quad - a_5b_2c_2 + a_6b_2c_1 + a_7b_1c_2 - a_8b_1c_1 \\
\eta_6 &:= a_2b_4c_3 - a_1b_4c_4 + a_3b_3c_4 - a_4b_3c_3 \\
&\quad - a_5b_2c_4 - a_5b_4c_2 + a_6b_2c_3 + a_6b_4c_1 \\
&\quad + a_7b_1c_4 + a_7b_3c_2 - a_8b_1c_3 - a_8b_3c_1 \\
\eta_7 &:= a_6b_4c_3 - a_5b_4c_4 + a_7b_3c_4 - a_8b_3c_3 \\
\eta_8 &:= -(a_1 + a_4) \\
\eta_9 &:= -(a_5 + a_8) \\
\eta_{10} &:= a_1a_4 - a_2a_3 \\
\eta_{11} &:= a_1a_8 - a_2a_7 - a_3a_6 + a_4a_5 \\
\eta_{12} &:= a_5a_8 - a_6a_7
\end{aligned}$$

Since either $B_1 = 0$ or $C_1 = 0$, we have $\eta_3 = \eta_7 = 0$. Since the algorithm converges to an optimal fixed point, we have from Lemma 4.4 that $G_\lambda(z)$ must have a zero at $z = 1$ for all nonzero eigenvalues λ of $\mathcal{L}$, so the term $(z - 1)$

must factor from the numerator. Therefore, the parameters satisfy $\eta_1 + \eta_4 = 0$, $\eta_2 + \eta_5 = 0$, and $\eta_6 = 0$. Also, the transfer function must have a pole at $z = 1$ when $\lambda = 0$, which requires the denominator to be $(z-1)^2$ when $\lambda = 0$. This implies $\eta_8 = -2$ and $\eta_{10} = 1$. The transfer function then has the form

$$G_\lambda(z) = \frac{(\eta_1 + \eta_2\lambda)(z-1)}{(z-1)^2 + \lambda\left(\eta_{11} + \eta_9 z + \eta_{12}\lambda\right)}, \tag{A.4}$$

which is equivalent to (4.8) with $(\eta_1, \eta_2, \eta_9, \eta_{11}, \eta_{12}) \mapsto (-\alpha, \alpha\zeta_3, \zeta_1, \zeta_0 - \zeta_1, \zeta_2)$. Note that η_1 cannot be zero, since this would violate the necessary condition that the transfer function has a pole at $z = 1$ when $\lambda = 0$. We can then invert the mapping to obtain the parameters $(\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3) = \left(-\eta_1, \eta_9 + \eta_{11}, \eta_9, \eta_{12}, -\frac{\eta_2}{\eta_1}\right)$. Therefore, the algorithm can be put into canonical form with a suitable choice of parameters $\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3$, and these parameters are unique since the mapping to the transfer function coefficients is one-to-one. Furthermore, there are five degrees of freedom in the coefficients of the transfer function (A.4), so any fully expressive canonical form must have at least five parameters. Our canonical form has precisely five parameters and is therefore a minimal parameterization. It remains to show that the technical conditions hold.

T1. Suppose $\alpha = 0$. Then the transfer function of the canonical form is identically zero, and therefore does not satisfy the conditions of Lemma 4.4. But this contradicts that the algorithm converges to an optimal fixed point, so $\alpha \neq 0$. Therefore, T1 holds.

T2. For a given initial state, let $(v_{1i}^\star, v_{2i}^\star, y_i^\star, u_i^\star, x_i^\star, w_i^\star)$ for $i \in \{1, \ldots, n\}$ denote the optimal fixed point to which the algorithm converges. Then this point must satisfy the linear equation (A.3). For this to hold for a general objective function, the system must have a solution for any $u_i^\star$ such that $\sum_{i=1}^n u_i^\star = 0$, which implies T2 holds.

T3. Since the algorithm converges to an optimal fixed point, the right side of (A.2) must be zero, or equivalently, $\zeta_0 \sum_{i=1}^n w_i^\star = 0$. The Laplacian matrix is symmetric, so $\sum_{i=1}^n w_i^{k+1} = \sum_{i=1}^n w_i^k$ for all $k \geq 0$. Therefore, we must have either $\zeta_0 = 0$ or $\sum_{i=1}^n w_i^0 = 0$, so T3 holds.

A.4 Proof of Theorem 5.1

Substituting the template (5.1) into the consensus LMI (3.10a) reduces to

$$P_{11}\begin{bmatrix}1-\rho^2 & -\alpha\\ -\alpha & \alpha^2\end{bmatrix}+M_0\preceq 0,$$

which is satisfied with $\alpha=(1-\rho)/m$ and $P_{11}=\frac{m\,(L-m)}{\rho\,(1-\rho)}$. Note that this LMI is known to describe the convergence rate of centralized gradient descent; see [13, Section 4.4].

Now consider the potential solution to the disagreement LMI (3.10b) given by

$$Q=\frac{t_3}{\alpha^2\rho^2}\begin{bmatrix}1+\rho^2\frac{t_1}{t_4} & -1\\ -1 & 1\end{bmatrix}\quad\text{and}\quad R=\frac{t_5}{\alpha^2 t_2},\quad\text{where}$$

$$t_1:=2\,(1-\beta)-\eta, \qquad t_2:=\beta-1+\rho^2,$$

$$t_3:=\beta\,(\eta+2\rho^2)-\eta\,(1-\rho^2), \qquad t_4:=2\beta\rho^2-\eta\,(1-\rho^2),$$

$$t_5:=(1-\beta-\rho)\big(\beta\,(\eta+2\rho^2)-(1-\rho^2)(1-\kappa+2\kappa\rho)\big),$$

$$t_6:=\big(2-\alpha(L+m)\big)(1-\rho^2)^2-\big(2(1-\rho^4)-\alpha(L+m)\big)\beta.$$

Using these values along with the value for σ^2 in (5.3b), the matrix in the disagreement LMI (3.10b) is equal to the rank-one matrix $-\frac{1}{t_2t_4}zz^{\mathsf{T}}$, where

$$z:=\frac{1}{\alpha\rho}\begin{bmatrix}t_6\\ -t_2t_3\\ \alpha\,t_2\,\big(2-\alpha\,(L+m)\big)\\ \beta\,\big(t_3-\alpha\rho^2(L+m)\big)\end{bmatrix}.$$

In order for this to be a valid solution, we must have $t_3>0$ and $t_1/t_4>0$ (so that $Q\succ 0$), $t_5/t_2\geq 0$ (so that $R\succeq 0$), and $t_2t_4>0$ (so that (3.10b) holds). All of these inequalities hold if and only if (5.3a) holds. Therefore, the SDP has a rank-one solution using the parameters in (5.2) if β and ρ satisfy (5.3). The convergence bound then follows from Theorem 3.10.