A Canonical Form for First-Order Distributed Optimization Algorithms

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

Abstract

We consider the distributed optimization problem in which a network of agents aims to minimize the average of local functions. To solve this problem, several algorithms have recently been proposed wherein agents perform various combinations of communication with neighbors, local gradient computations, and updates to local state variables. In this paper, 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.

1 Introduction

We consider the distributed optimization problem in which a network of n agents are connected through an undirected graph. The global objective is to minimize the average of n functions $f_i : \mathbb{R}^d \to \mathbb{R}$ that are local to each agent $i = 1, \dots, n$. While each agent has access to its local variable x_i, it must exchange information with its neighbors to arrive at the global minimizer

$$x^\star := \operatorname*{arg\,min}_{x \in \mathbb{R}^d} f(x), \quad \text{where } f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x). \tag{1}$$

Distributed optimization is an active area of research and has attracted significant attention in recent years due to its applications in distributed machine learning, distributed estimation, and resource allocation [1, 5, 7, 10]. To solve (1), many gradient-based algorithms have been proposed involving gradient computation steps in conjunction with gossip steps, i.e., local averaging operations [6, 8, 9, 15, 18, 20, 21]. Such 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 [6] algorithm update:

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

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 [8]. In contrast, Exact Diffusion [20, 21] uses two state variables which are updated with a gradient step followed by a gossip step:

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

While their update equations appear different, we show in Section 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 class of first-order distributed algorithms.

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

P1. *The algorithm is linear, time-invariant, deterministic, and all agents perform the same updates.*

P2. *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.*

P3. *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.*

While NIDS and Exact Diffusion belong to the class of algorithms described by P1–P3, not all algorithms do. Exceptions include accelerated [12, 17], proximal [16], stochastic [11], and asynchronous [19] variants.

[1]A. Sundararajan and L. Lessard are with the Department of Electrical and Computer Engineering at the University of Wisconsin–Madison, Madison, WI 53706, USA.

[2]B. Van Scoy and L. Lessard are with the Wisconsin Institute for Discovery, which is also at the University of Wisconsin–Madison. {asundararaja,vanscoy,laurent.lessard}@wisc.edu

This material is based upon work supported by the National Science Foundation under Grant No. 1656951.

1.1 Canonical form

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 $L \in \mathbb{R}^{n\times n}$ to model local communication. This is equivalent to using a gossip matrix W with $W := I - L$. We make the following assumption throughout the paper.

Assumption 1. *The Laplacian matrix $L \in \mathbb{R}^{n\times n}$ is symmetric, positive semidefinite, connected, and satisfies $L\mathbf{1} = 0$ where $\mathbf{1}$ is the all-ones vector.*

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

Canonical Form

Initialization: Let $L \in \mathbb{R}^{n\times n}$ 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

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

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

    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 $\{x_i^k\}$ 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 $\alpha u = (\zeta_0 I + \zeta_2 L)w$ 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 $w_i^0 = 0$.*

1.2 Main results

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

Definition 2 (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 $i \in \{1,\dots,n\}$ such that $y_i^\star = x^\star$ for all $i \in \{1,\dots,n\}$, where $x^\star$ is defined in* (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 5.1.

Theorem 3 (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. 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 5.2.

Theorem 4 (Necessity)**.** *Consider a distributed algorithm that satisfies properties* P1–P3 *and converges to an optimal fixed point. Then there exists 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 2 and provide associated necessary conditions that hold when the algorithm converges to an optimal fixed point. We then show how to put distributed algorithms into canonical form through an example in Section 3.

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 which solves (1). These conditions are used to prove Theorem 4, and also give rise to an impossibility result stating that no single-state algorithm can solve the distributed optimization problem.

Consider the following general dynamical system model for a distributed optimization algorithm.[1]

$$\begin{bmatrix}\xi_i^{k+1}\\ y_i^k\end{bmatrix} = \left(\begin{bmatrix}A_0 & B_0\\ C_0 & D_0\end{bmatrix}\otimes I_d\right)\begin{bmatrix}\xi_i^k\\ u_i^k\end{bmatrix} + \left(\begin{bmatrix}A_1 & B_1\\ C_1 & D_1\end{bmatrix}\otimes I_d\right)\sum_{j=1}^n L_{ij}\begin{bmatrix}\xi_j^k\\ u_j^k\end{bmatrix} \tag{2a}$$

$$u_i^k = \nabla f_i(y_i^k) \tag{2b}$$

[1] The iterations (2) are similar to the polynomial linear protocol [3] used to study the dynamic average consensus problem.

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 (2) 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 (2) 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

$$\xi^{k+1} = A(L)\xi^k + B(L)u^k \tag{3a}$$
$$y^k = C(L)\xi^k + D(L)u^k \tag{3b}$$
$$u^k = \nabla f(y^k) \tag{3c}$$

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

$$A(L) := I_n \otimes A_0 + L \otimes A_1, \quad B(L) := I_n \otimes B_0 + L \otimes B_1,$$
$$C(L) := I_n \otimes C_0 + L \otimes C_1, \quad D(L) := I_n \otimes D_0 + L \otimes D_1.$$

By Assumption 1, we may write $L = V\Lambda V^\mathsf{T}$ 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 $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_n$. For $\ell = 1, \dots, n$ we define the state, input, and output in direction v_ℓ as follows:

$$\bar{\xi}_\ell^k := (v_\ell^\mathsf{T} \otimes I_2)\xi^k \qquad \bar{u}_\ell^k := v_\ell^\mathsf{T} u^k \qquad \bar{y}_\ell^k := v_\ell^\mathsf{T} y^k \tag{4}$$

In these new coordinates, (3) is equivalent to the n decoupled single-input single-output systems

$$\bar{\xi}_\ell^{k+1} = (A_0 + \lambda_\ell A_1)\bar{\xi}_\ell^k + (B_0 + \lambda_\ell B_1)\bar{u}_\ell^k \tag{5a}$$
$$\bar{y}_\ell^k = (C_0 + \lambda_\ell C_1)\bar{\xi}_\ell^k + (D_0 + \lambda_\ell D_1)\bar{u}_\ell^k \tag{5b}$$
$$\bar{u}_\ell^k = v_\ell^\mathsf{T} \nabla f\big(v_1\, \bar{y}_1^k + \cdots + v_n\, \bar{y}_n^k\big) \tag{5c}$$

for all $\ell \in \{1, \dots, n\}$ where $\bar{\xi}_\ell^0 \in \mathbb{R}^2$. We refer to (5) as the *separated system*, and the corresponding transfer functions are given by

$$G_{\lambda_\ell}(z) = (C_0 + \lambda_\ell C_1)\big(zI - (A_0 + \lambda_\ell A_1)\big)^{-1}(B_0 + \lambda_\ell B_1) + (D_0 + D_1\lambda_\ell). \tag{6}$$

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 $G_{\lambda_\ell}(z)$ derived above, imposing the additional constraint that the algorithm has an optimal fixed point (Definition 2) leads to properties that all transfer functions must necessarily satisfy. We summarize these in the following lemma; see Appendix 5.3 for a proof.

Lemma 5. *Suppose algorithm* (2) *converges to an optimal fixed point. Then the transfer function* (6) *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 $\ell = 2, \dots, n$.*

The requirements on the transfer function in Lemma 5 imply that no single-state algorithm of the form (2) can solve the distributed optimization problem. This provides an explanation as to why the Distributed Gradient Descent algorithm must use a diminishing stepsize [9]. 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 6. *No algorithm satisfying properties* P1 *and* P3 *where each agent has a local state of dimension d can solve the distributed optimization problem* (1).

Proof of Corollary 6. Such an algorithm can be written in the form (2) 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

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

Suppose the algorithm solves the distributed optimization problem (1). Then from Lemma 5, $G_{\lambda_0}(z)$ 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. ∎

3 Putting algorithms in canonical form

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 (2) with

$$\left[\begin{array}{c|c} A_0 & B_0 \\ \hline C_0 & D_0 \end{array}\right] = \left[\begin{array}{cc|c} 1 & \zeta_0 & -\alpha \\ 0 & 1 & 0 \\ \hline 1 & 0 & 0 \end{array}\right] \quad \text{and}$$
$$\left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] = \left[\begin{array}{cc|c} -\zeta_1 & \zeta_2 & 0 \\ -1 & 0 & 0 \\ \hline -\zeta_3 & 0 & 0 \end{array}\right] \tag{7}$$

and the corresponding transfer function (6) is

$$G_\lambda^{\text{CF}}(z) = -\frac{\alpha\,(1 - \zeta_3\lambda)(z-1)}{(z-1)(z-1+\zeta_1\lambda) + \lambda\,(\zeta_0 + \zeta_2\lambda)}. \tag{8}$$

We can then apply the following procedure to put any distributed algorithm satisfying properties P1–P3 in our canonical form. We summarize the results for several known algorithms in Table 1.

1. Write the algorithm in the form (2) 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 $G_\lambda(z)$ using (6).
3. The parameters $\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3$ are then obtained by comparing coefficients of the transfer function $G_\lambda(z)$ with that of the canonical form $G_\lambda^{\text{CF}}(z)$ in (8).

Table 1: Parameters of distributed optimization algorithms in canonical form using $W = I - L$.

	ζ_0	ζ_1	ζ_2	ζ_3
EXTRA [15]	$\frac{1}{2}$	1	0	0
NIDS [6]	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Exact Diffusion [20, 21]	$\frac{1}{2}$	1	0	$\frac{1}{2}$
DIGing [8, 13]	0	2	1	0
AsynDGM^2 [19]	0	2	1	1
Jakovetić [4] ($\mathcal{B} = \beta I$)	$\alpha\beta$	2	1	0
Jakovetić [4] ($\mathcal{B} = \beta W$)	$\alpha\beta$	2	$1 - \alpha\beta$	0

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

Remark 7. *To provide an additional degree of freedom, we can relate the gossip matrix to the Laplacian matrix by $W = I - \mu L$ where $\mu \in \mathbb{R}$ is an additional scalar parameter. This is equivalent to choosing the gossip matrix as a convex combination of $I - L$ and the identity, i.e., $W = (1 - \mu)I + \mu(I - L)$.*

Remark 8 (Uniqueness of realization)**.** *The canonical form* (7) *can be reliably obtained from an algorithm by computing its transfer function* (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* (7)*:*

$$\left[\begin{array}{c|c} A_0 & B_0 \\ \hline C_0 & D_0 \end{array}\right] = \left[\begin{array}{cc|c} 1 & \zeta_0 & -\alpha \\ 0 & 1 & 0 \\ \hline 1 & 0 & 0 \end{array}\right] \quad \textit{and}$$

$$\left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] = \left[\begin{array}{cc|c} -\zeta_1 & \zeta_2 & \alpha\zeta_3 \\ -1 & 0 & 0 \\ \hline 0 & 0 & 0 \end{array}\right].$$

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 [2, 14].

3.1 Example: NIDS

To illustrate our approach, we return to the NIDS algorithm from Section 1. As suggested by the authors of [6], we choose $\widetilde{W} := (I + W)/2$ where the gossip matrix W is related to the Laplacian matrix L by $W = I - L$.

First, we write NIDS in the form (2) as follows:

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

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

This is equivalent to (2) with state-space matrices

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

$$\left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] = \left[\begin{array}{ccc|c} -1 & \frac{1}{2} & -\frac{\alpha}{2} & \frac{\alpha}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right]$$

which have associated transfer function (6) given by

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

Comparing coefficients with that of $G_\lambda^{\mathrm{CF}}(z)$ in (8), we find that the parameters of the canonical form are $(\zeta_0, \zeta_1, \zeta_2, \zeta_3) = (\frac{1}{2}, 1, 0, \frac{1}{2})$.

4 Conclusion

In this paper, we derived a canonical form for a large class of first-order distributed algorithms that fulfills existence and uniqueness properties, and features a minimal parametrization. We provided sufficient conditions for an algorithm in canonical form to have a fixed point corresponding to the solution of the distributed optimization problem, as well as necessary conditions for convergence to such a fixed point. This provides a first step toward a principled and automated methodology for the analysis and design of distributed optimization algorithms.

References

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

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

[3] R. Freeman, T. Nelson, and K. 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.

[2]The similar algorithm AugDGM [18] does not fit in our framework because it requires two sequential rounds of communication per iteration.

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

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

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

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

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

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

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

[11] S. Pu and A. Nedić. A distributed stochastic gradient tracking method. *arXiv preprint arXiv:1803.07741*, 2018.

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

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

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

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

[16] W. Shi, Q. Ling, G. Wu, and W. Yin. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63(22):6013–6023, 2015.

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

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

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

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

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

5 Appendix

5.1 Proof of Theorem 3

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 $i \in \{1, \dots, n\}$. 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 $y_i^\star = x_i^\star$ from (C.3), and $x_i^\star = x_j^\star$ for all $i, j \in \{1, \dots, n\}$ since the graph is connected by Assumption 1. 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). To do so, we sum (C.5) to obtain

$$\alpha \sum_{i=1}^n u_i^\star = \zeta_0 \sum_{i=1}^n w_i^\star - \zeta_1 \sum_{i,j=1}^n L_{ij} x_j^\star + \zeta_2 \sum_{i,j=1}^n L_{ij} w_j^\star. \quad (9)$$

The last two terms on the right side are both zero since the Laplacian matrix is symmetric by Assumption 1, and therefore satisfies $L^\mathsf{T}\mathbf{1} = 0$. 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) and is therefore optimal.

An optimal fixed point exists. Define $x^\star$ as in (1). We show how to construct an optimal fixed point of the canonical form. Define $v_{1i}^\star := 0$, $y_i^\star := x^\star$, $u_i^\star := \nabla f_i(x^\star)$, and $x_i^\star := x^\star$ for all $i \in \{1, \dots, n\}$. We have $\sum_{i=1}^n u_i^\star = 0$ from the definition of $x^\star$, so the linear system

$$\alpha u_i^\star = \zeta_0 w_i^\star + \zeta_2 \sum_{j=1}^n L_{ij}\, w_j^\star \quad (10)$$

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

5.2 Proof of Theorem 4

Consider an algorithm that satisfies properties P1–P3 and converges to an optimal fixed point. Such an algorithm can be represented by the iterations (2) 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

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

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 5 that $G_\lambda(z)$ must have a zero at $z = 1$ for all nonzero eigenvalues λ of 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

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

which is equivalent to (8) with the mapping $(\eta_1, \eta_2, \eta_9, \eta_{11}, \eta_{12}) \mapsto (-\alpha, \alpha\zeta_3, \zeta_1, \zeta_0 - \zeta_1, \zeta_2)$. 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 $(\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3) = \big(-\eta_1, \eta_9 + \eta_{11}, \eta_9, \eta_{12}, -\frac{\eta_2}{\eta_1}\big)$. 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 (11), 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 (8) is identically zero, and therefore does not satisfy the conditions of Lemma 5. 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 $i \in \{1, \dots, n\}$ denote the optimal fixed point to which the algorithm converges. Then this point must satisfy the linear equation (10). 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 (9) must be zero, or equivalently, $\zeta_0 \sum_{i=1}^n w_i^\star = 0$. The Laplacian matrix is symmetric by Assumption 1, 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. ∎

5.3 Proof of Lemma 5

Since the algorithm (2) converges to an optimal fixed point, the separated systems (5) also converge to fixed points, which satisfy $\bar{y}_i^\star = x^\star$ by Definition 2. When $\ell = 1$, we have $\lambda_1 = 0$ and $v_1 = \frac{1}{\sqrt{n}}\mathbf{1}$. Using (4) and (1),

$$\bar{u}_1^\star = v_1^\mathsf{T} u^\star = \tfrac{1}{\sqrt{n}} \sum_{i=1}^n \nabla f_i(y_i^\star) = \sqrt{n}\, \nabla f(x^\star) = 0. \tag{12a}$$

For $\ell = 2, \dots, n$, each v_ℓ is orthogonal to v_1 since L is symmetric. Again using (4) and (1),

$$\bar{y}_\ell^\star = v_\ell^\mathsf{T} y^\star = v_\ell^\mathsf{T} \mathbf{1} x^\star = 0 \qquad \text{for } \ell \in \{2, \dots, n\}. \tag{12b}$$

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 (12a), 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, $G_{\lambda_1}(z)$ must be marginally stable and in particular must have 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 (12b). 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 (12b) holds. ∎