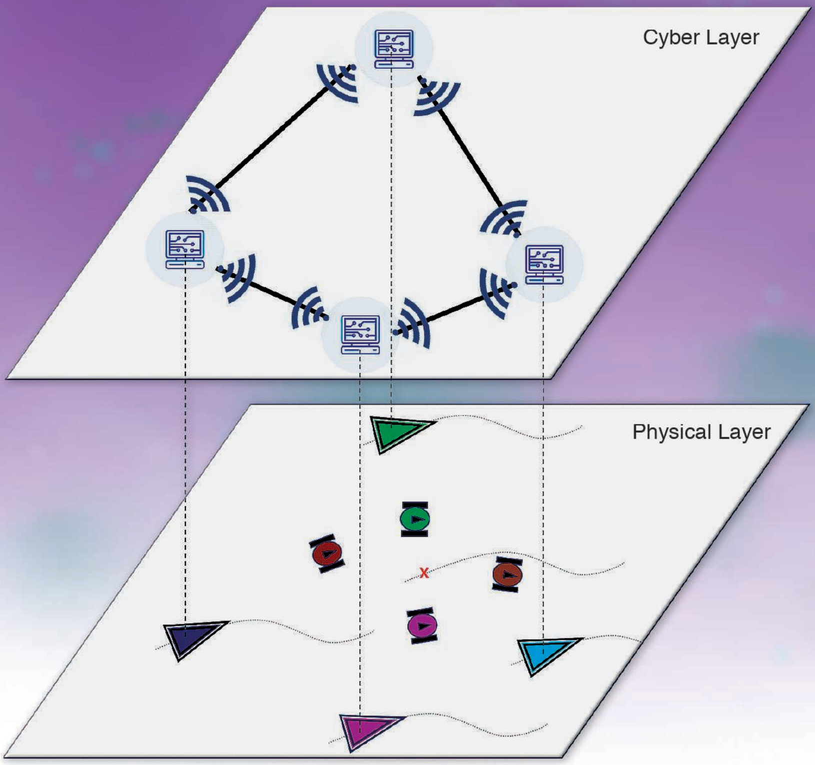


Tutorial on Dynamic Average Consensus

THE PROBLEM, ITS APPLICATIONS, AND THE ALGORITHMS

SOLMAZ S. KIA, BRYAN VAN SCOY, JORGE CORTÉS, RANDY A. FREEMAN, KEVIN M. LYNCH, and SONIA MARTÍNEZ

Technological advances in ad hoc networking and the availability of low-cost reliable computing, data storage, and sensing devices have made scenarios possible where the coordination of many subsystems extends the range of human capabilities. Smart grid operations, smart transportation, smart health care, and sensing networks for environmental monitoring and exploration in hazardous situations are just a few examples of such network operations. In these applications, the ability of a network system to (in a decentralized fashion) fuse information, compute common estimates of unknown quantities, and agree on a common view of the world is critical. These problems can be formulated as agreement problems on linear combinations of dynamically changing reference signals or local parameters. This dynamic agreement problem corresponds to *dynamic average consensus*, which, as discussed in "Summary," is the problem of interest of this article. The dynamic average consensus problem is for a group of agents to cooperate to track the average of locally available time-varying reference signals, where each agent is capable only of local computations and communicating with local neighbors (see Figure 1).

Digital Object Identifier 10.1109/MCS.2019.2900783
Date of publication: 17 May 2019

1066-033X/19©2019IEEE

CENTRALIZED SOLUTIONS HAVE DRAWBACKS

The difficulty of the dynamic average consensus problem is that the information is distributed across the network. A straightforward solution, termed *centralized*, to the dynamic average consensus problem is to gather all of the information in a single place, perform the computation (in other words, calculate the average), and then send the solution back through the network to each agent. Although simple, the centralized approach has numerous drawbacks: 1) the algorithm is not robust to failures of the centralized agent (if the centralized agent fails, then the entire computation fails), 2) the method is not scalable because the amount of communication and memory required on each agent scales with the size of the network, 3) each agent must have a unique identifier (so that the centralized agent counts each value only once), 4) the calculated average is delayed by an amount that grows with the size of the network, and 5) the reference signals from each agent are exposed over the entire network (which is unacceptable in applications involving sensitive data). The centralized solution is fragile due to the existence of a single failure point in the network. This can be overcome by having every agent act as the centralized agent. In this approach, referred to as *flooding*, agents transmit the values of the reference signals across the entire network until each agent knows each reference signal. This may be summarized as first do all communications and then do all computations. While flooding fixes the issue of robustness to agent failures, it is still subject to many of the drawbacks of the centralized solution. Although this approach works reasonably well for small-size networks, its communication and storage costs scale poorly in terms of the network size and may incur, depending on how it is implemented, costs of order $O(N^2)$ per agent (for instance, this is the case if each agent maintains which neighbors it has or has not sent each piece of information to). This motivates the interest in developing distributed solutions for the dynamic average consensus problem that involve only local interactions and decisions among the agents.

CHALLENGES WITH DYNAMIC PROBLEMS

The static version of the dynamic average consensus problem (commonly referred to as static average consensus) is the familiar problem in which agents seek to agree on a specific combination of fixed quantities. The static problem has been extensively studied in the literature [1]–[4], and several simple and efficient distributed algorithms exist with exact convergence guarantees. Given its mature literature, a natural approach to address the distributed solution of the dynamic average consensus problem in some literature has been to zero-order sample the reference signals and use a static average consensus algorithm between sampling times (for example, see [5] and [6]). If this was a practical approach, it would mean that there is no need to worry about designing specific algorithms to solve the dynamic average consensus problem because we could rely on the algorithmic solutions available for static average consensus.

However, this approach does not work because it would essentially need a static average consensus algorithm that is able to converge infinitely fast. In practice, some time is required for information to flow across the network, and hence the result of the repeated application of any static average consensus algorithm operates with some error whose size depends on its speed of convergence and how fast inputs change. To illustrate this point better, we have

Summary

This article addresses the dynamic average consensus problem and the distributed coordination algorithms available to solve it. Such a problem arises in scenarios with multiple agents, where each one has access to a time-varying signal of interest (for example, a robot sensor sampling the position of a mobile target of interest or a distributed energy resource taking a sequence of frequency measurements in a microgrid). The dynamic average consensus problem consists of having the multiagent network collectively compute the average of the set of time-varying signals. Reasons for pursuing this objective are numerous and include data fusion, refinement of uncertainty guarantees, and computation of higher-accuracy estimates, all enabling local decision making with network-wide aggregated information. Solving this problem is challenging because the local interactions among agents involve only partial information, and the quantity that the network seeks to compute is changing as the agents run their routines. The article provides a tutorial introduction to distributed methods that solve the dynamic average consensus problems, paying special attention to the role of network connectivity and incorporating information about the nature of the time-varying signals, the performance tradeoffs regarding convergence rate, steady-state error and memory and communication requirements, and algorithm robustness against initialization errors.

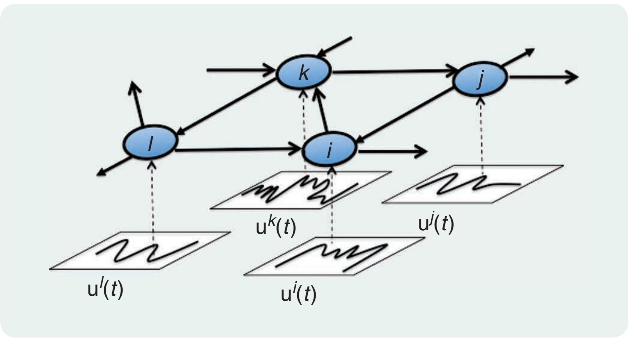


FIGURE 1 A group of communication agents, each endowed with a time-varying reference signal.

the following numerical example. Consider a process described by a fixed value plus a sine wave whose frequency and phase are changing randomly over time. A group of six agents with the communication topology of a directed ring monitors this process by taking synchronous samples, each according to

$$\mathsf{u}^i(m) = a^i(2 + \sin(\omega(m)t(m) + \phi(m))) + b^i, \quad m \in \mathbb{Z}_{\geq 0},$$

where a^i and b^i are fixed unknown error sources in the measurement of agent $i \in \{1,\ldots,6\}$. To reduce the effect of measurement errors, after each sampling, every agent wants the average of the measurements across the network before the next sampling time. For the numerical simulations, the values $\omega \sim N(0,0.25)$, $\phi \sim N(0,(\pi/2)^2)$, with $N(\mu,p)$ indicating a Gaussian distribution with mean μ and variance p, are used. The sampling rate is set to 0.5 Hz ($\Delta t = 2$ s). For the simulation under study, $a^1 = 1.1, a^2 = 1$, $a^3 = 0.9$, $a^4 = 1.05$, $a^5 = 0.96$, $a^6 = 1$, $b^1 = -0.55$, $b^2 = 1$, $b^3 = 0.6, b^4 = -0.9, b^5 = -0.6$, and $b^6 = 0.4$. To obtain the average, the folllowing two approaches are used: 1) at every sampling time m, each agent initializes the standard static discrete-time Laplacian average consensus algorithm

$$x^i(k+1) = x^i(k) - \delta \sum_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)), \qquad i \in \{1,\ldots,N\},$$

by the current sampled reference values $x^i(0) = \mathsf{u}^i(m)$ and implements it with an admissible time step δ until just before the next sampling time $m+1$; 2) at time $m = 0$, agents start executing a dynamic average consensus algorithm [more specifically, strategy (S15), which is described in detail later]. Between sampling times m and $m+1$, the reference input $\mathsf{u}^i(k)$ implemented in the algorithm is fixed at $\mathsf{u}^i(m)$, where k is the communication time index. Figure 2 compares the tracking performance of these two approaches. It is observed that the dynamic average consensus algorithm, by keeping a memory of past actions, produces a better tracking response than the static algorithm initialized at each sampling time with the current values. This comparison serves as motivation for the need to specifically design distributed algorithms that take into account the particular features of the dynamic average consensus problem.

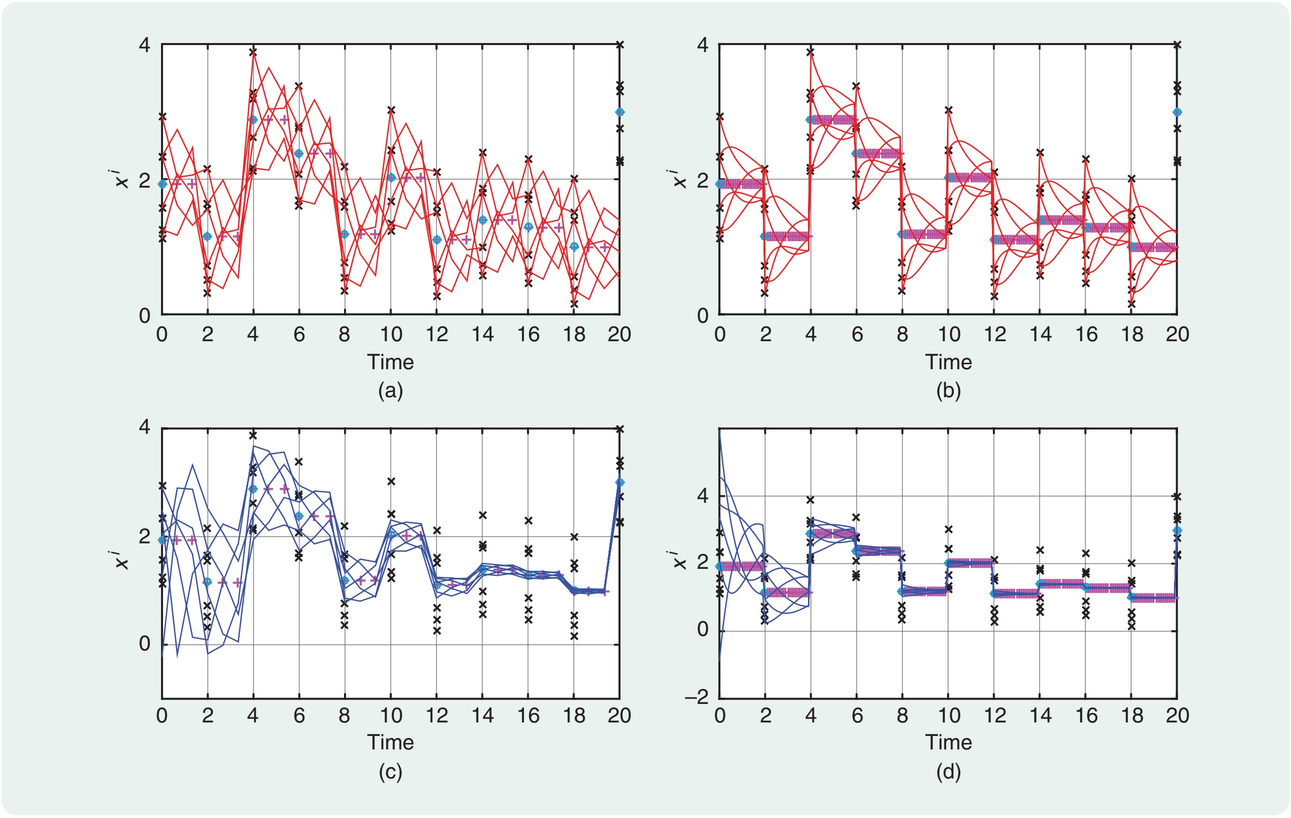


FIGURE 2 A comparison of performance between a static average consensus algorithm reinitialized at each sampling time versus a dynamic average consensus algorithm. The solid lines: red curves (respectively, blue curves) represent the time history of the agreement state of each agent generated by the Laplacian static average consensus approach [respectively, the dynamic average consensus of (S15)]; $\times$: sampling points at $m\Delta t$; ○: the average at $m\Delta t$; $+$: the average of reference signals at $k\delta$. The dynamic consensus algorithm very closely tracks the average over time as the static consensus does not have enough time between sampling times to converge. This trend is preserved even if the frequency of the communication between the agents increases. In these simulations, $\alpha = \beta = 1$ in (S15). (a) Static algorithm; three communications in $t \in [m, m+1]$. (b) Static algorithm; 20 communications in $t \in [m, m+1]$. (c) Dynamic algorithm; three communications in $t \in [m, m+1]$. (d) Dynamic algorithm; 20 communications in $t \in [m, m+1]$.

OBJECTIVES AND ARTICLE ROAD MAP

The objective of this article is to provide an overview of the dynamic average consensus problem that serves as a comprehensive introduction to the problem definition, its applications, and the distributed methods available to solve them. This article was motivated by the fact that, in the literature, many works exist that have dealt with the problem. However, there is not a tutorial reference that presents, in a unified way, the developments that have occurred over the years. "Summary" encapsulates the contents of the article, emphasizing the value and utility of its algorithms and results. The primary intention, rather than providing a full account of all of the available literature, is to introduce, in a tutorial fashion, the main ideas behind dynamic average consensus algorithms, the performance tradeoffs considered in their design, and the requirements needed for their analysis and convergence guarantees.

The article first introduces the problem definition and a set of desired properties expected from a dynamic average consensus algorithm. Next, various applications of dynamic average consensus in network systems are presented, including distributed formation, distributed state estimation, and distributed optimization problems. It is not surprising that the initial synthesis of dynamic average consensus algorithms emerged from a careful look at static average consensus algorithms. The section "A Look at Static Average Consensus Leading up to the Design of a Dynamic Average Consensus Algorithm" provides a brief review of standard algorithms for the static average consensus and then builds on this discussion to describe the first dynamic average consensus algorithm. Various features of these initial algorithms are elaborated on, and their shortcomings are identified. This sets the stage in the section "Continuous-Time Dynamic Average Consensus Algorithms" to introduce various algorithms that address these shortcomings. The design of continuous-time algorithms for network systems is often motivated by the conceptual ease for design and analysis, rooted in the relatively mature theoretical basis for the control of continuous-time systems. However, the implementation of these continuous-time algorithms on cyber-physical systems may not be feasible due to practical constraints, such as limited interagent communication bandwidth. This motivates the section "Discrete-Time Dynamic Average Consensus Algorithms," which specifically discusses methods to accelerate the convergence rate and enhance the robustness of the proposed algorithms. Because the information of each agent takes some time to propagate through the network, it is expected that tracking an arbitrarily fast average signal with zero error is not feasible unless agents have some a priori information about the dynamics generating the signals. This topic is addressed in the "Perfect Tracking Using A Priori Knowledge of the Input Signals" section, which takes advantage of knowledge of the nature of the reference signals. Many other topics exist that are related to the dynamic average consensus problem not explored in this article. Several intriguing pointers for such topics are in "Further Reading." Throughout the article, unless otherwise noted, network systems are considered whose communication topology is described by strongly connected and weight-balanced directed graphs. In only a few specific cases, the discussion focuses on the setup of undirected graphs, and these are explicitly mentioned.

REQUIRED MATHEMATICAL BACKGROUND AND AVAILABLE RESOURCES FOR IMPLEMENTATION

Graph theory plays an essential role in the design and performance analysis of dynamic consensus algorithms. "Basic Notions from Graph Theory" provides a brief overview of the relevant graph theoretic concepts, definitions, and notations in this article. Dynamic average consensus algorithms are linear time-invariant (LTI) systems in which the reference signals of the agents enter the system as an external input, in contrast to the (Laplacian) static average consensus algorithm, where the reference signals enter as initial conditions. Thus, in addition to the internal stability analysis (which is sufficient for the static average consensus algorithm), the input-to-state stability (ISS) of the algorithms must be assessed. A brief overview of the ISS analysis of LTI systems is provided in "Input-to-State Stability of Linear Time-Invariant Systems." All of the algorithms described can be implemented using such modern computing languages as C and Matlab. Matlab provides functions for the simple construction, modification, and visualization of graphs.

DYNAMIC AVERAGE CONSENSUS: PROBLEM FORMULATION

Consider a group of N agents where each agent is capable of 1) sending and receiving information with other agents, 2) storing information, and 3) performing local computations. For example, the agents may be cooperating robots or sensors in a wireless sensor network. The communication topology among the agents is described by a fixed digraph (see "Basic Notions from Graph Theory"). Suppose that each agent has a local scalar reference signal, denoted $\mathsf{u}^i(t) : [0, \infty) \to \mathbb{R}$ in continuous time and $\mathsf{u}^i(k) : \mathbb{N} \to \mathbb{R}$ in discrete time. This signal may be the output of a sensor located on the agent, or it could be the output of another algorithm that the agent is running. The dynamic average consensus problem then consists of designing an algorithm that allows individual agents to track the time-varying average of the reference signals, given by

$$\text{continuous time:} \quad \mathsf{u}^{\text{avg}}(t) := \frac{1}{N}\sum_{i=1}^{N} \mathsf{u}^i(t),$$

$$\text{discrete time:} \quad \mathsf{u}^{\text{avg}}(k) := \frac{1}{N}\sum_{i=1}^{N} \mathsf{u}^i(k).$$

For discrete-time signals and algorithms, for any variable p sampled at time t_k, the shorthand notation $p(k)$ or p_k is used to refer to $p(t_k)$. For reasons specified later, the design of distributed algorithms is of specific interest, meaning that to obtain the average, the policy that each agent implements depends only on its variables (represented by J^i, which includes its own reference signal) and those of its out-neighbors (represented by $\{I^j\}_{j \in \mathcal{N}^i_{\text{out}}}$).

In continuous time, a *driving command* $c^i(J^i(t), \{I^j(t)\}_{j \in \mathcal{N}^i_{\text{out}}}) \in \mathbb{R}$ is sought for each agent $i \in \{1,\dots,N\}$ such that (with an appropriate initialization) a local state $x^i(t)$ (which is referred to as the *agreement state* of agent i) converges to the average $\mathsf{u}^{\text{avg}}(t)$ asymptotically. Formally, for

$$\text{continuous time:} \quad \dot{x}^i = c^i(J^i(t), \{I^j(t)\}_{j \in \mathcal{N}^i_{\text{out}}}), \quad i \in \{1,\dots,N\}, \tag{1}$$

Further Reading

Numerous works have studied the robustness of dynamic average consensus algorithms against a variety of disturbances and sources of error present in practical scenarios. These include fixed communication delays [S1], additive input disturbances [S2], time-varying communication graphs [S3], and driving command saturation [19]. Variations of the dynamic average consensus problems explore scenarios where the algorithm design depends on the specific agent dynamics [S4], [S5], [71] or incorporates different agent roles, such as in leader–follower networks of mobile agents [15], [S6], [S7].

When dealing with directed agent interactions, a common assumption in solving the average consensus problem is that the communication graph is *weight balanced*, which is equivalent to the graph consensus matrix $\mathbf{W} := \mathbf{I} - \mathbf{L}$ being doubly stochastic. In [S8], it is shown that calculating an average over a network requires either explicit or implicit use of either 1) the out-degree of each agent, 2) global node identifiers, 3) randomization, or 4) asynchronous updates with specific properties. In particular, the balanced assumption is necessary for scalable, deterministic, synchronous algorithms. In general, agents may not have access to their out-degree (for example, agents that use local broadcast communication). If each agent knows its out-degree, however, then distributed algorithms may be used to generate weight-balanced and doubly stochastic digraphs [S9], [S10].

Another approach is to explicitly use the out-degree in the algorithm by having agents share their out-weights and use them to adjust for the imbalances in the graph. This approach is referred to as the *push-sum* protocol and has been applied to the static average consensus problem (see [S11]–[S14]). Both of these approaches of dealing with unbalanced graphs require each agent to know its out-degree. Furthermore, when communication links are time varying, these approaches work only if the time varying graph remains weight balanced (see [19] and [S15]). If communication failures caused by limited communication ranges or external events, such as obstacle blocking, destroy the weight-balanced character of the graph, then it is still possible to solve the dynamic average consensus problem if the expected graph is balanced [S3]. Another set of works has explored the question of how to optimize the graph topology to endow consensus algorithms with better properties. These include designing the network topology in the presence of random link failures [S16] and optimizing the edge weights for fast consensus [S17], [7].

REFERENCES

[S1] H. Moradian and S. S. Kia, "On robustness analysis of a dynamic average consensus algorithm to communication delay," *IEEE Trans. Control Netw. Syst.* Aug. 6, 2018. doi: 10.1109/TCNS.2018.2863568.
[S2] G. Shi and K. H. Johansson, "Robust consensus for continuous-time multi-agent dynamics," *SIAM J. Control Optim.*, vol. 51, no. 5, pp. 3673–3691, 2013.
[S3] B. Van Scoy, R. A. Freeman, and K. M. Lynch, "Asymptotic mean ergodicity of average consensus estimators," in *Proc. American Control Conf.*, 2014, pp. 4696–4701.
[S4] F. Chen, G. Feng, L. Liu, and W. Ren, "Distributed average tracking of networked Euler–Lagrange systems," *IEEE Trans. Autom. Control*, vol. 60, no. 2, pp. 547–552, 2015.
[S5] S. Ghapania, W. Ren, F. Chen, and Y. Song, "Distributed average tracking for double-integrator multi-agent systems with reduced requirement on velocity measurements," *Automatica*, vol. 81, no. 7, pp. 1–7, 2017.
[S6] G. Shi, Y. Hong, and K. H. Johansson, "Connectivity and set tracking of multi-agent systems guided by multiple moving leaders," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 663–676, 2012.
[S7] Z. Meng, D. V. Dimarogonas, and K. H. Johansson, "Leader-follower coordinated tracking of multiple heterogeneous Lagrange systems using continuous control," *IEEE Trans. Robot.*, vol. 30, no. 3, pp. 739–745, 2014.
[S8] J. M. Hendrickx and J. N. Tsitsiklis, "Fundamental limitations for anonymous distributed systems with broadcast communications," in *Proc. Allerton Conf. Communication, Control, and Computing*, 2015, pp. 9–16.
[S9] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *Eur. J. Control*, vol. 18, no. 6, pp. 539–557, 2012.
[S10] A. Rikos, T. Charalambous, and C. N. Hadjicostis, "Distributed weight balancing over digraphs," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 2, pp. 190–201, 2014.
[S11] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proc. IEEE Int. Symp. Information Theory*, 2010, pp. 1753–1757.
[S12] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed matrix scaling and application to average consensus in directed graphs," *IEEE Trans. Autom. Control*, vol. 58, no. 3, pp. 667–681, 2013.
[S13] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, 2015.
[S14] P. Rezaienia, B. Gharesifard, T. Linder, and B. Touri. Push-sum on random graphs. 2017. [Online]. Available: arXiv:1708.00915
[S15] S. S. Kia, J. Cortés, and S. Martínez, "Distributed event-triggered communication for dynamic average consensus in networked systems," *Automatica*, vol. 59, pp. 112–119, Sept. 2015.
[S16] S. Kar and J. M. F. Moura, "Sensor networks with random links: Topology design for distributed consensus," in *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3315–3326, 2008.
[S17] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *Proc. IEEE Conf. Decision and Control*, 2003, pp. 4997–5002.

with proper initialization if necessary, $x^i(t) \to \mathsf{u}^{\text{avg}}(t)$ as $t \to \infty$. The driving command c^i can be a memoryless function or an output of a local internal dynamics. Note that, by using the out-neighbors, the convention is made that information flows in the opposite direction specified by a directed edge (there is no loss of generality in doing it so, and the alternative convention of using in-neighbors instead would also be equally valid).

Dynamic average consensus can also be accomplished using discrete-time dynamics, especially when the time-varying inputs are sampled at discrete times. In such a case, a *driving command* is sought for each agent $i \in \{1,\dots,N\}$ so that

$$\text{discrete time:}\quad x^i(t_{k+1}) = c^i(J^i(t_k), \{I^j(t_k)\}_{j\in\mathcal{N}^i_{\text{out}}}),\quad i\in\{1,\dots,N\}, \tag{2}$$

under proper initialization if necessary, accomplishes $x^i(t_k) \to \mathsf{u}^{\text{avg}}(k)$ as $t_k \to \infty$. Algorithm 1 illustrates how a discrete-time dynamic average consensus algorithm can be executed over a network of N communicating agents.

Also, consider a third class of dynamic average consensus algorithms in which the dynamics at the agent level are in continuous time, but the communication among the agents, because of the restrictions of the wireless communication devices, takes place in discrete time:

$$\text{continuous time–discrete time:}\quad \dot{x}^i(t) = c^i(J^i(t), \{I^j(t^j_{k^j})\}_{j\in\mathcal{N}^i_{\text{out}}}),\quad i\in\{1,\dots,N\}, \tag{3}$$

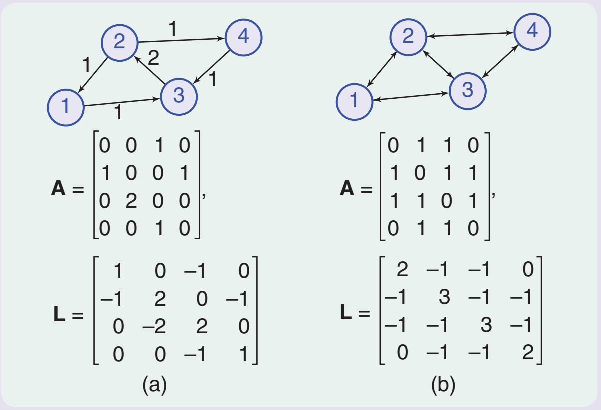
such that $x^i(t) \to \mathsf{u}^{\text{avg}}(t)$ as $t \to \infty$. Here, $t^j_{k^j} \in \mathbb{R}$ is the k^jth transmission time of agent j, which is not necessarily

Basic Notions From Graph Theory

The communication network of a multiagent cooperative system can be modeled by a *directed graph*, or *digraph*. Here, we briefly review some basic concepts from graph theory following [S18]. A digraph is a pair $\mathcal{G} = (\mathcal{V},\mathcal{E})$, where $\mathcal{V} = \{1,\dots,N\}$ is the *node set* and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ is the *edge set*. An edge from i to j, denoted by (i, j), means that agent j can send information to agent i. For an edge $(i, j) \in \mathcal{E}$, i is called an *in-neighbor* of j, and j is called an *out-neighbor* of i. We denote the set of out-neighbors of each agent i by $\mathcal{N}^i_{\text{out}}$. A graph is *undirected* if $(i, j) \in \mathcal{E}$ any time $(j, i) \in \mathcal{E}$ (see Figure S1).

A *weighted digraph* is a triplet $\mathcal{G} = (\mathcal{V},\mathcal{E},\mathbf{A})$, where $(\mathcal{V},\mathcal{E})$ is a digraph and $\mathbf{A} \in \mathbb{R}^{N\times N}$ is a weighted *adjacency* matrix with the property that $a_{ij} > 0$ if $(i, j) \in \mathcal{E}$ and $a_{ij} = 0$, otherwise. A weighted digraph is *undirected* if $a_{ij} = a_{ji}$ for all $i, j \in \mathcal{V}$. The *weighted out-degree* and *weighted in-degree* of a node i are, respectively, $\mathrm{d}^{\text{out}}(i) = \Sigma_{j=1}^N a_{ji}$ and $\mathrm{d}^{\text{in}}(i) = \Sigma_{j=1}^N a_{ji}$. Let $\mathrm{d}^{\text{out}}_{\max} = \max_{i\in\{1,\dots,N\}} \mathrm{d}^{\text{out}}(i)$ denote the maximum weighted out-degree. A digraph is *weight balanced* if, at each node $i \in \mathcal{V}$, the weighted out-degree and weighted in-degree coincide (although they might be different across different nodes). The out-degree matrix $\mathbf{D}^{\text{out}}$ is the diagonal matrix with entries $\mathbf{D}^{\text{out}}_{ii} = \mathrm{d}^{\text{out}}(i)$, for all $i \in \mathcal{V}$. The *(out-) Laplacian* matrix is $\mathbf{L} = \mathbf{D}^{\text{out}} - \mathbf{A}$. Note that $\mathbf{L}\mathbf{1}_N = 0$. A weighted digraph $\mathcal{G}$ is weight balanced if and only if $\mathbf{1}_N^\top \mathbf{L} = 0$. Based on the structure of $\mathbf{L}$, at least one of the eigenvalues of $\mathbf{L}$ is zero and the rest of them have nonnegative real parts. Denote the eigenvalues of $\mathbf{L}$ by $\lambda_i, i \in \{1,\dots,N\}$, where $\lambda_1 = 0$ and $\Re(\lambda_i) \le \Re(\lambda_j)$, for $i < j$. For strongly connected digraphs, $\operatorname{rank}(\mathbf{L}) = N - 1$. For strongly connected and weight-balanced digraphs, denote the eigenvalues of $\operatorname{Sym}(\mathbf{L}) = (\mathbf{L} + \mathbf{L}^\top)/2$ by $\hat{\lambda}_1,\dots,\hat{\lambda}_N$, where $\hat{\lambda}_1 = 0$ and $\hat{\lambda}_i \le \hat{\lambda}_j$, for $i < j$. For strongly connected and weight-balanced digraphs,

$$0 < \hat{\lambda}_2 \mathbf{I} \le \mathbf{R}^\top \operatorname{Sym}(\mathbf{L})\mathbf{R} \le \hat{\lambda}_N \mathbf{I}, \tag{S1}$$



$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 0 & 2 & 0 & 0\\ 0 & 0 & 1 & 0 \end{bmatrix},\qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & -1 & 0\\ -1 & 2 & 0 & -1\\ 0 & -2 & 2 & 0\\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0\\ 1 & 0 & 1 & 1\\ 1 & 1 & 0 & 1\\ 0 & 1 & 1 & 0 \end{bmatrix},\qquad \mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0\\ -1 & 3 & -1 & -1\\ -1 & -1 & 3 & -1\\ 0 & -1 & -1 & 2 \end{bmatrix}$$

FIGURE S1 Examples of directed and undirected graphs. (a) Strongly connected, weight-balanced digraph. (b) Connected graph with unit edge weights.

where $\mathbf{R} \in \mathbb{R}^{N\times(N-1)}$ satisfies $[(1/\sqrt{N})\mathbf{1}_N\ \ \mathbf{R}][(1/\sqrt{N})\mathbf{1}_N\ \ \mathbf{R}]^\top = [(1/\sqrt{N})\mathbf{1}_N\ \ \mathbf{R}]^\top[(1/\sqrt{N})\mathbf{1}_N\ \ \mathbf{R}] = \mathbf{I}_N$. Note that for connected graphs, $\operatorname{Sym}(\mathbf{L}) = \mathbf{L}$, and consequently $\lambda_i = \hat{\lambda}_i$, for all $i \in \mathcal{V}$.

Intuitively, the Laplacian matrix can be viewed as a diffusion operator over the graph. To illustrate this, suppose each agent $i \in \mathcal{V}$ has a scalar variable $x^i \in \mathbb{R}$. Stacking the variables into a vector $\mathbf{x}$, multiplication by the Laplacian matrix gives the weighted sum

$$[\mathbf{L}\mathbf{x}]_i = \sum_{j\in\mathcal{V}} a_{ij}(x^i - x^j), \tag{S2}$$

where a_{ij} is the weight of the link between agents i and j.

REFERENCE

[S18] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks* (Applied Mathematics Series). Princeton, NJ: Princeton Univ. Press, 2009.

Input-to-State Stability of Linear Time-Invariant Systems

For a linear time-invariant (LTI) system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad \mathbf{x} \in \mathbb{R}^n, \; \mathbf{u} \in \mathbb{R}^m, \tag{S3}$$

the solution for $t \in \mathbb{R}_{\geq 0}$ can be written as

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\,d\tau. \tag{S4}$$

For a Hurwitz matrix $\mathbf{A}$, by using the bound

$$\| e^{\mathbf{A}t} \| \leq \kappa e^{-\underline{\lambda} t}, \quad t \in \mathbb{R}_{\geq 0}, \tag{S5}$$

for some $\kappa, \underline{\lambda} \in \mathbb{R}_{>0}$, an upper bound on the norm of the trajectories of (S4) is established as

$$\begin{aligned} \| \mathbf{x}(t) \| &\leq \kappa e^{-\underline{\lambda} t} \| \mathbf{x}(0) \| + \int_0^t \kappa e^{-\underline{\lambda}(t-\tau)} \| \mathbf{B} \| \| \mathbf{u}(\tau) \| d\tau \\ &\leq \kappa e^{-\underline{\lambda} t} \| \mathbf{x}(0) \| + \frac{\kappa \| \mathbf{B} \|}{\underline{\lambda}} \sup_{0 \leq \tau \leq t} \| \mathbf{u}(\tau) \|, \quad \forall t \in \mathbb{R}_{\geq 0}. \end{aligned} \tag{S6}$$

The bound shows that the zero-input response decays to zero exponentially fast, whereas the zero-state response is bounded for every bounded input, indicating an input-to-state stability behavior. Note that the ultimate bound on the system state is proportional to the bound on the input.

Next, how to compute the parameters $\kappa, \underline{\lambda} \in \mathbb{R}_{>0}$ is shown in (S5). Recall that [S19, Fact 11.15.5] for any matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. Therefore,

$$\| e^{\mathbf{A}t} \| \leq e^{\lambda_{\max}(\mathrm{Sym}(\mathbf{A}))t}. \quad \forall t \in \mathbb{R}_{\geq 0}, \tag{S7}$$

where $\mathrm{Sym}(\mathbf{A}) = (1/2)(\mathbf{A} + \mathbf{A}^\top)$. Therefore, for a Hurwitz matrix $\mathbf{A}$ whose $\mathrm{Sym}(\mathbf{A})$ is also Hurwitz, the exponential bound parameters can be set to

$$\underline{\lambda} = -\lambda_{\max}(\mathrm{Sym}(\mathbf{A})), \quad \kappa = 1. \tag{S8}$$

A tighter exponential bound of

$$\underline{\lambda} = \lambda^\star, \quad \kappa = \sqrt{\sigma_{\max}(\mathbf{P}^\star)/\sigma_{\min}(\mathbf{P}^\star)}, \tag{S9}$$

can also be obtained for any Hurwitz system matrix A, according to [S20, Prop. 5.5.33], from the convex linear matrix inequality optimization problem

$$(\lambda^\star, \mathbf{P}^\star) = \operatorname{argmin} \lambda \quad \text{subject to} \tag{S10a}$$

$$\mathbf{P}\mathbf{A} + \mathbf{A}^\top \mathbf{P} \leq -2\lambda \mathbf{P}, \quad \mathbf{P} > 0, \quad \lambda > 0. \tag{S10b}$$

Similarly, the state of the discrete-time, LTI system

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \quad \mathbf{x}_k \in \mathbb{R}^n, \; \mathbf{u}_k \in \mathbb{R}^m \tag{S11}$$

with initial condition $\mathbf{x}_0 \in \mathbb{R}^n$ satisfies the bound

$$\| \mathbf{x}_k \| \leq \sqrt{\frac{\sigma_{\max}(\mathbf{P})}{\sigma_{\min}(\mathbf{P})}} \left(\rho^k \| \mathbf{x}_0 \| + \frac{1-\rho^k}{1-\rho} \| \mathbf{B} \| \sup_{0 \leq j < k} \| \mathbf{u}_j \| \right), \tag{S12}$$

where $\mathbf{P} \in \mathbb{R}^{n \times n}$ and $\rho \in \mathbb{R}$ satisfy

$$\mathbf{A}^\top \mathbf{P}\mathbf{A} - \rho^2 \mathbf{P} \leq 0, \quad \mathbf{P} > 0, \quad \rho \geq 0. \tag{S13}$$

REFERENCES

[S19] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear System Theory*. New York: Springer-Verlag, 2005.

[S20] D. Hinrichsen and A. J. Pritchard, *Mathematical Systems Theory I: Modeling, State Space Analysis, Stability and Robustness*. Princeton, NJ: Princeton Univ. Press, 2005.

ALGORITHM 1 The execution of a discrete-time dynamic average consensus algorithm at each agent $i \in \{1, \ldots, N\}$.

```
Input: $J^i(k)$ and $\{I^j(k)\}_{j \in \mathcal{N}^i_{\mathrm{out}}}$
Output: $x^i(k+1)$, $J^i(k+1)$, and $I^i(k+1)$
Step 1. $x^i(k+1) \leftarrow c^i(J^i(t_k), \{I^j(t_k)\}_{j \in \mathcal{N}^i_{\mathrm{out}}})$
Step 2. Generate $J^i(k+1)$ and $I^i(k+1)$
Step 3. Broadcast $I^i(k+1)$
```

synchronous with the transmission time of other agents in the network.

The consideration of simple dynamics of the form in (1)–(3) is motivated by the fact that the state of the agents does not necessarily correspond to some physical quantity but, instead, to some logical variable on which agents perform computation and processing. Agreement on the average is also of relevance in scenarios where the agreement state is a physical state with more complex dynamics, for example, the position of a mobile agent in a robotic team. In such cases, this discussion can be leveraged by, for instance, having agents compute reference signals that are to be tracked by the states with more complex dynamics. See "Further Reading" for a list of relevant literature on dynamic average consensus problems for higher-order dynamics. Given the drawbacks of centralized solutions, several desirable properties when designing algorithmic solutions to the dynamic average consensus problem are identified:

- *scalability*, so that the amount of computations and resources required on each agent does not grow with the network size
- *robustness* to the disturbances present in practical scenarios, such as communication delays and packet drops, agents entering/leaving the network, and noisy measurements

» *correctness*, meaning the algorithm converges to the exact average or, alternatively, a formal guarantee can be given about the distance between the estimate and the exact average.

Regarding the last property, to achieve agreement, network connectivity must be such that information about the local reference input of each agent reaches other agents frequently enough. As the information of each agent takes some time to propagate through the network, tracking an arbitrarily fast average signal with zero error is not feasible unless agents have some a priori information about the dynamics generating the signals. A recurring theme throughout the article is how the convergence guarantees of dynamic average consensus algorithms depend on the network connectivity and rate of change of the reference signal of each agent.

APPLICATIONS OF DYNAMIC AVERAGE CONSENSUS IN NETWORK SYSTEMS

The ability to compute the average of a set of time-varying reference signals is useful in numerous applications, which explains why distributed algorithmic solutions have found their way into many seemingly different problems involving the interconnection of dynamical systems. This section provides a selected overview of problems to motivate further research on dynamic average consensus algorithms and illustrate their range of applicability. Other applications of dynamic average consensus can be found in [7]–[13].

Distributed Formation Control

Autonomous networked mobile agents are playing an increasingly important role in coverage, surveillance, and patrolling applications in both commercial and military domains. The tasks accomplished by mobile agents often require dynamic motion coordination and formation among team members. Consensus algorithms have been commonly used in the design of formation control strategies [14]–[16]. These algorithms have been used, for instance, to arrive at agreement on the geometric center of formation so that the formation can be achieved by spreading the agents in the desired geometry about this center (see [1]). However, most of the existing results are for static formations. Dynamic average consensus algorithms can effectively be used in dynamic formation control, where quantities of interest such as the geometric center of the formation change with time. Figure 3 depicts an example scenario in which a group of mobile agents tracks a team of mobile targets. Each agent monitors a mobile target with location $\mathbf{x}_T^i$. The objective is for the agents to follow the team of mobile targets by spreading out in a prespecified formation, which consists of each agent being positioned at a relative vector $\mathbf{b}^i$ with respect to the time-varying geometric center of the target team. A two-layer approach can be used to accomplish the formation and tracking objectives in this scenario: a dynamic consensus algorithm in the cyber layer that computes the geometric center in a distributed manner and a physical layer that tracks this average plus $\mathbf{b}^i$. Note that dynamic average consensus algorithms can also be employed to compute the time-varying variance of the positions of the mobile targets with respect to the geometric center, and this can help the mobile agents adjust the scale of the formation to avoid collisions with the target team. Examples of the use of dynamic consensus algorithms

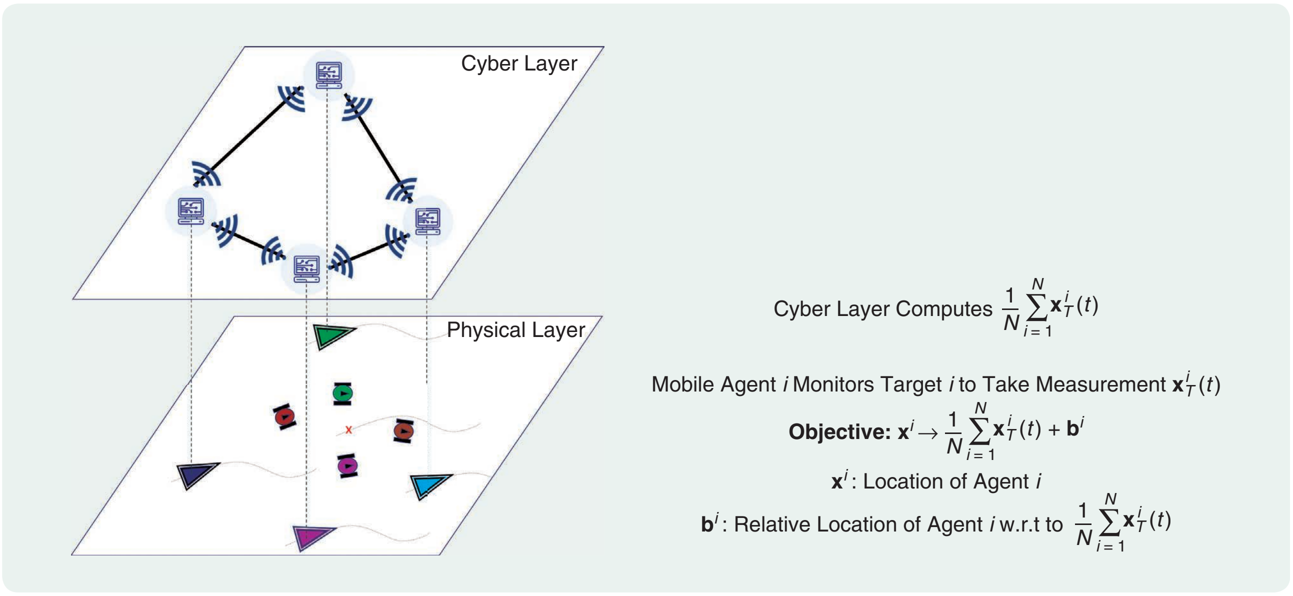

Cyber Layer
Physical Layer
Cyber Layer Computes $\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_T^i(t)$
Mobile Agent i Monitors Target i to Take Measurement $\mathbf{x}_T^i(t)$
Objective: $\mathbf{x}^i \to \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_T^i(t) + \mathbf{b}^i$
$\mathbf{x}^i$: Location of Agent i
$\mathbf{b}^i$: Relative Location of Agent i w.r.t to $\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_T^i(t)$

FIGURE 3 A two-layer consensus-based formation for tracking a team of mobile targets. The larger triangle robots are the mobile agents, and the smaller round robots are the mobile moving targets. The physical layer shows the situational distribution of the mobile agents and the moving targets. The cyber layer shows which mobile agent has a computational capability and the interagent communication topology.

in this two-layer approach with multiagent systems with first-order, second-order, or higher-order dynamics can be found in [17]–[19].

Distributed State Estimation

Wireless sensors with embedded computing and communication capabilities play a vital role in provisioning real-time monitoring and control in many applications, such as environmental monitoring, fire detection, object tracking, and body area networks. Consider a model of the process of interest given by

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\boldsymbol{\omega}(k),$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, $\mathbf{A}(k) \in \mathbb{R}^{n\times n}$ and $\mathbf{B}(k) \in \mathbb{R}^{n\times m}$ are known system matrices, and $\boldsymbol{\omega} \in \mathbb{R}^m$ is the white Gaussian process noise with $E[\boldsymbol{\omega}(k)\boldsymbol{\omega}^\top(k)] = \mathbf{Q} > 0$. Let the measurement model at each sensor station $i \in \{1,\ldots,N\}$ be

$$\mathbf{z}^i(k+1) = \mathbf{H}^i(k+1)\mathbf{x}(k+1) + \boldsymbol{\nu}^i,$$

where $\mathbf{z}^i \in \mathbb{R}^q$ is the measurement vector, $\mathbf{H}^i \in \mathbb{R}^{q\times n}$ is the measurement matrix, and $\boldsymbol{\nu}^i \in \mathbb{R}^q$ is the white Gaussian measurement noise with $E[\boldsymbol{\nu}^i(k)\boldsymbol{\nu}^i(k)^\top] = \mathbf{R}^i > 0$. If all of the measurements are transmitted to a fusion center, a Kalman filter can be used to obtain the minimum variance estimate of the state of the process of interest as (see Figure 4):

» propagation stage

$$\hat{\mathbf{x}}^-(k+1) = \mathbf{A}(k)\hat{\mathbf{x}}^-(k), \tag{4a}$$
$$\mathbf{P}^-(k+1) = \mathbf{A}(k)\mathbf{P}^-(k+1)\mathbf{A}(k)^\top + \mathbf{B}(k)\mathbf{Q}(k)\mathbf{B}(k)^\top, \tag{4b}$$
$$\mathbf{Y}^-(k+1) = \mathbf{P}^-(k+1)^{-1}, \tag{4c}$$
$$\mathbf{y}^-(k+1) = \mathbf{Y}^-(k+1)\hat{\mathbf{x}}^-(k+1); \tag{4d}$$

» update stage

$$\begin{aligned}
\mathbf{Y}^i(k+1) &= \mathbf{H}^i(k+1)^\top\mathbf{R}^i(k+1)^{-1}\mathbf{H}^i(k+1), \quad i \in \{1,\ldots,N\},\\
\mathbf{y}^i(k+1) &= \mathbf{H}^i(k+1)^\top\mathbf{R}^i(k+1)^{-1}\mathbf{z}^i(k+1), \quad i \in \{1,\ldots,N\},\\
\mathbf{P}^+(k+1) &= \left(\mathbf{Y}^-(k+1) + \sum_{i=1}^{N}\mathbf{Y}^i(k+1)\right)^{-1},\\
\hat{\mathbf{x}}^+(k+1) &= \mathbf{P}^+(k+1)\left(\mathbf{y}^-(k+1) + \sum_{i=1}^{N}\mathbf{y}^i(k+1)\right).
\end{aligned}$$

Despite its optimality, this implementation is not desirable in many sensor network applications due to the existence of a single point of failure at the fusion center and the high cost of communication between the sensor stations and the fusion center. An alternative that has previously gained interest [5], [20]–[23] is to employ distributed algorithmic solutions that have each sensor station maintain a local filter to process its local measurements and fuse them with the estimates of its neighbors. Some work [20], [24], [25] employs dynamic average consensus to synthesize distributed implementations of the Kalman filter. For instance, one of the early solutions for distributed minimum variance estimation has each agent maintain a local copy of the propagation filter (4) and employ a dynamic average consensus algorithm to generate the coupling time-varying terms $(1/N)\Sigma_{i=1}^N \mathbf{y}^i(k+1)$ and $(1/N)\Sigma_{i=1}^N \mathbf{Y}^i(k+1)$. If agents know the size of the network, then they can duplicate the update equation locally.

Distributed Unconstrained Convex Optimization

The control literature has introduced numerous distributed algorithmic solutions [26]–[33] to solve unconstrained convex optimization problems over networked systems. In a distributed unconstrained convex optimization problem, a group of N communicating agents, each with access to a local convex cost function $f^i : \mathbb{R}^n \to \mathbb{R}, i \in \{1,\ldots,N\}$, seeks

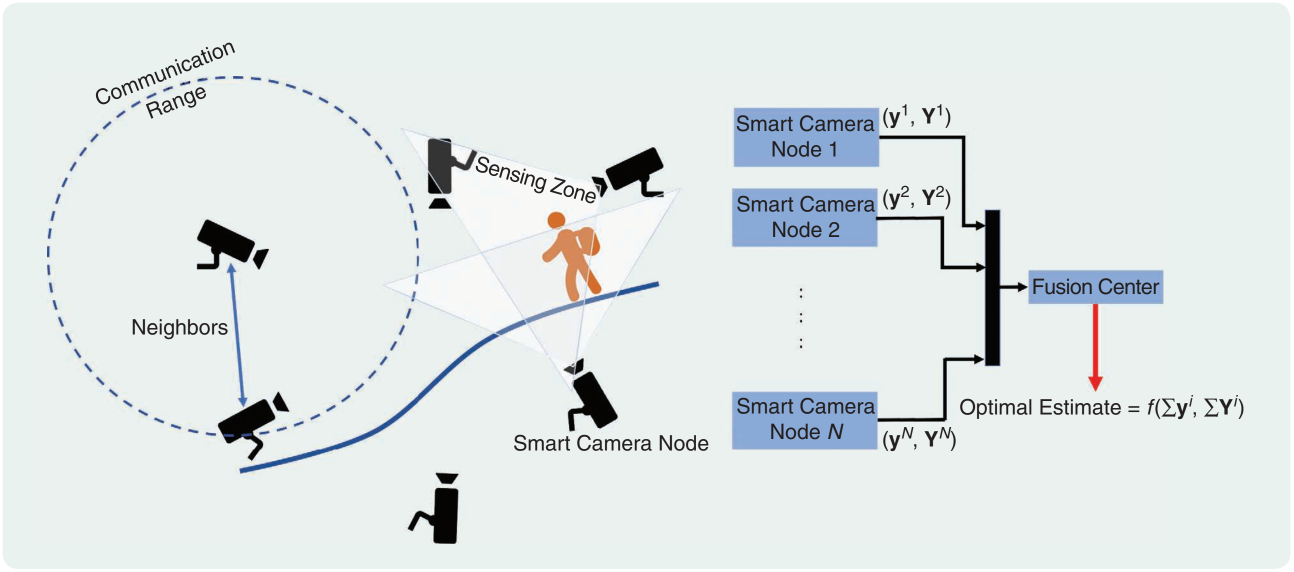


FIGURE 4 A networked smart camera system that monitors and estimates the position of moving targets.

to determine the minimizer of the joint global optimization problem

$$\mathbf{x}^{\star} = \operatorname{argmin} \frac{1}{N} \sum_{i=1}^{N} f^{i}(\mathbf{x}) \tag{5}$$

by local interactions with their neighboring agents. This problem appears in network system applications, such as multiagent coordination, distributed state estimation over sensor networks, or large-scale machine-learning problems. Some of the algorithmic solutions for this problem are developed using agreement algorithms to compute global quantities that appear in existing centralized algorithms. For example, a centralized solution for (5) is the Nesterov gradient descent algorithm [34] described by

$$\mathbf{x}(k+1) = \mathbf{y}(k) - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\nabla f^{i}(\mathbf{y}(k))\right), \tag{6a}$$

$$\mathbf{v}(k+1) = \mathbf{y}(k) - \frac{\eta}{\alpha_k}\left(\frac{1}{N}\sum_{i=1}^{N}\nabla f^{i}(\mathbf{y}(k))\right), \tag{6b}$$

$$\mathbf{y}(k+1) = (1-\alpha_{k+1})\mathbf{x}(k+1) + \alpha_{t+1}\mathbf{v}(k+1), \tag{6c}$$

where $\mathbf{x}(0), \mathbf{y}(0), \mathbf{v}(0) \in \mathbb{R}^{n}$, and $\{\alpha_k\}_{k=0}^{\infty}$ are defined by an arbitrarily chosen $\alpha_0 \in (0,1)$ and the update equation $\alpha_{k+1}^{2} = (1-\alpha_{k+1})\alpha_k^{2}$, where α_{k+1} always takes the unique solution in (0, 1). If all $f^{i}, i \in \{1,\dots,N\}$, are convex, differentiable, and have L-Lipschitz gradients, then every trajectory $k \mapsto \mathbf{x}(k)$ of (6) converges to the optimal solution $\mathbf{x}^{\star}$ for any $0 < \eta < (1/L)$.

Note that in (6), the cumulative gradient term $(1/N)\Sigma_{i=1}^{N} \nabla f^{i}(\mathbf{y}(k))$ is a source of coupling among the computations performed by each agent. It does not seem reasonable to halt the execution of this algorithm at each step until the agents have determined the value of this term. Instead, dynamic average consensus can be employed in conjunction with (6): the dynamic average consensus algorithm estimates the coupling term, and this estimate is employed in executing (6), which in turn changes the value of the coupling term being estimated. This approach is taken in [33] to solve the optimization problem (5) over connected graphs and is also pursued in other implementations of distributed convex or nonconvex optimization algorithms (see, for instance, [35]–[39]).

Distributed Resource Allocation

In optimal resource allocation, a group of agents works cooperatively to meet a demand in an efficient way (see Figure 5). Each agent incurs a cost for the resources it provides. Let the cost of each agent $i \in \{1,\dots,N\}$ be modeled by a convex and differentiable function $f^{i}:\mathbb{R} \to \mathbb{R}$. The objective is to meet the demand $d \in \mathbb{R}$ so that the total cost $f(\mathbf{x}) = \Sigma_{i=1}^{N} f^{i}(x^{i})$ is minimized. Each agent $i \in \{1,\dots,N\}$, therefore, seeks to find the ith element of $\mathbf{x}^{\star}$ given by

$$\mathbf{x}^{\star} = \operatorname{argmin}_{\mathbf{x}\in\mathbb{R}^{N}} \sum_{i=1}^{N} f^{i}(x^{i}), \text{ subject to } x^{1} + \dots + x^{N} - d = 0.$$

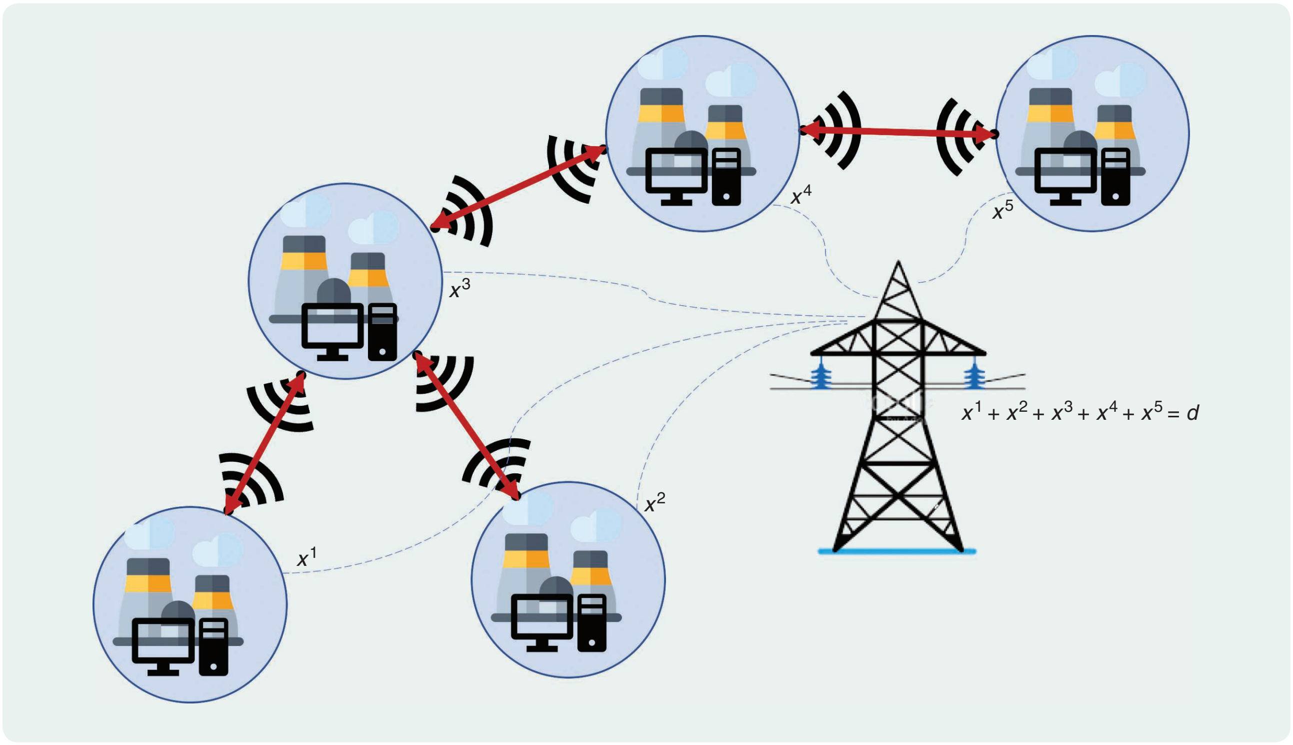


FIGURE 5 A network of five generators with connected undirected topology works together to meet a demand of $x^1 + x^2 + x^3 + x^4 + x^5 = d$ in a manner in which the overall cost $\Sigma_{i=1}^{5} f^{i}(x^{i})$ for the group is minimized.

This problem appears in many optimal decision-making tasks, such as optimal dispatch in power networks [40], [41], optimal routing [42], and economic systems [43]. For instance, the group of agents could correspond to a set of flexible loads in a microgrid that receive a request from a grid operator to collectively adjust their power consumption to provide a desired amount of regulation to the bulk grid. In this demand-response scenario, x^i corresponds to the amount of deviation from the nominal consumption of load i, the function f_i models the amount of discomfort caused by deviating from it, and d is the amount of regulation requested by the grid operator.

A centralized algorithmic solution is given by the popular saddle point or primal-dual dynamics [44], [45] associated to the optimization problem,

$$\dot{\mu}(t) = x^1(t) + \cdots + x^N(t) - d, \qquad \mu(0) \in \mathbb{R}, \tag{7a}$$

$$\dot{x}^i(t) = -\nabla f^i(x^i(t)) - \mu(t), \;\; i \in \{1, \ldots, N\}, \quad x^i(0) \in \mathbb{R}. \tag{7b}$$

If the local cost functions are strictly convex, then every trajectory $t \mapsto \mathbf{x}(t)$ converges to the optimal solution $\mathbf{x}^\star$. The source of coupling in (7) is the demand mismatch that appears in the right-hand side of (7a). However, the dynamic average consensus can be employed to estimate this quantity online and feed it back into the algorithm. This approach is taken in [46] and [47]. This can be accomplished, for instance, by having agent i use the reference signal $x^i(t) - d/N$ (this assumes that every agent knows the demand and number of agents in the network, but other reference signals are also possible) in a dynamic consensus algorithm coupled with the execution of (7).

A LOOK AT STATIC AVERAGE CONSENSUS LEADING UP TO THE DESIGN OF A DYNAMIC AVERAGE CONSENSUS ALGORITHM

Consensus algorithms to solve the static average consensus problem have been studied since [48]. The commonality in their design is the idea of having agents start their agreement state with their own reference value and adjust it based on some weighted linear feedback, which takes into account the difference between their agreement state and their neighbors'. This leads to algorithms of the form

$$\text{continuous time:} \quad \dot{x}^i(t) = -\sum_{j=1}^{N} a_{ij}(x^i(t) - x^j(t)), \tag{8a}$$

$$\text{discrete time:} \quad x^i(k+1) = x^i(k) - \sum_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)), \tag{8b}$$

for $i \in \{1, \ldots, N\}$, with $x^i(0) = \mathsf{u}^i$ constant for both algorithms. Here, $[a_{ij}]_{N \times N}$ is the adjacency matrix of the communication graph (see "Basic Notions from Graph Theory"). By stacking the agent variables into vectors, the static average consensus algorithms can be written compactly using a graph Laplacian as

$$\text{continuous time:} \quad \dot{\mathbf{x}}(t) = -\mathbf{L}\mathbf{x}(t), \tag{9a}$$

$$\text{discrete time:} \quad \mathbf{x}(k+1) = (\mathbf{I} - \mathbf{L})\mathbf{x}(k), \tag{9b}$$

with $\mathbf{x}(0) = \mathsf{u}$. When the communication graph is fixed, this system is LTI and can be analyzed using standard time-domain and frequency-domain techniques in control. Specifically, the frequency-domain representation of the static average consensus algorithm output signal is given by

$$\text{continuous time:} \quad \mathbf{X}(s) = [s\mathbf{I} + \mathbf{L}]^{-1}\mathbf{x}(0) = [s\mathbf{I} + \mathbf{L}]^{-1}\mathbf{U}(s), \tag{10a}$$

$$\text{discrete time:} \quad \mathbf{X}(z) = [z\mathbf{I} - (\mathbf{I} - \mathbf{L})]^{-1}\mathbf{U}(z), \tag{10b}$$

where $\mathbf{X}(s)$ and $\mathbf{U}(s)$, respectively, denote the Laplace transform of $\mathbf{x}(t)$ and $\mathbf{u}$, while $\mathbf{X}(z)$ and $\mathbf{U}(z)$, respectively, denote the z-transform of $\mathbf{X}_k$ and $\mathbf{u}$. For static signals, $\mathbf{U}(s) = \mathbf{u}$ and $\mathbf{U}(z) = \mathbf{u}$.

The block diagram of these static average consensus algorithms is shown in Figure 6. The dynamics of these algorithms consists of a negative feedback loop, where the feedback term is composed of the Laplacian matrix and an integrator [$1/s$ in continuous time and $1/(z-1)$ in discrete time]. For the static average consensus algorithms, the reference signal enters the system as the initial condition of the integrator state. Under certain conditions on the communication graph, the error of these algorithms can be shown to converge to zero, as stated next.

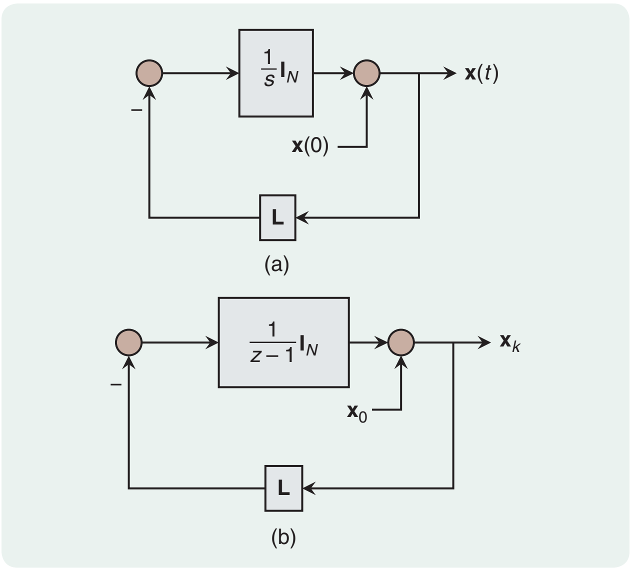


FIGURE 6 A block diagram of the static average consensus algorithms (9). The input signals are assigned to the initial conditions, that is, $\mathbf{x}(0) = \mathbf{u}$ (in continuous time) or $\mathbf{x}_0 = \mathbf{u}$ (in discrete time). The feedback loop consists of the Laplacian matrix of the communication graph and an integrator [$1/s$ in continuous time and $1/(z-1)$ in discrete time]. (a) Continuous time. (b) Discrete time.

Theorem 1: Convergence Guarantees of the Continuous-Time and Discrete-Time Static Average Consensus Algorithms (8) [1]

Suppose that the communication graph is a constant, strongly connected, and weight-balanced digraph and that the reference signal u^i at each agent $i \in \{1,\dots,N\}$ is a constant scalar. Then the following convergence results hold for the continuous-time and discrete-time static average consensus algorithms (8):

» continuous time: As $t \to \infty$, every agreement state $x^i(t), i \in \{1,\dots,N\}$ of the continuous-time static average consensus algorithm (8a) converges to u^{avg} with an exponential rate no worse than $\hat{\lambda}_2$, the smallest nonzero eigenvalue of $\mathrm{Sym}(\mathbf{L})$.

» discrete time: As $k \to \infty$, every agreement state $x_k^i, i \in \{1,\dots,N\}$ of the discrete-time static average consensus algorithm (8b) converges to u^{avg} with an exponential rate no worse than $\rho \in (0,1)$, provided that the Laplacian matrix satisfies $\rho = \|\mathbf{I}_N - \mathbf{L} - \mathbf{1}_N\mathbf{1}_N^\top/N\|_2 < 1$. □

Note that, given a weighted graph with Laplacian matrix $\mathbf{L}$, the graph weights can be scaled by a nonzero constant $\delta \in \mathbb{R}$ to produce a scaled Laplacian matrix $\delta\mathbf{L}$ (see "Basic Notions from Graph Theory"). This extra scaling parameter can then be used to produce a Laplacian matrix that satisfies the conditions in Theorem 1.

A First Design for Dynamic Average Consensus

Because the reference signals enter the static average consensus algorithms (8) as initial conditions, they cannot track time-varying signals. Looking at the frequency-domain representation in Figure 6 of the static average consensus algorithms (8), it is clear that what is needed instead is to continuously inject the signals as inputs into the dynamical system. This allows the system to naturally respond to changes in the signals without any need for reinitialization. This basic observation is made in [49], resulting in the systems shown in Figure 7.

More precisely, [49] argues that considering the static inputs as a dynamic step function, the algorithm

$$\dot{\mathbf{x}}(t) = -\mathbf{L}x(t) + \dot{\mathbf{u}}(t), \quad x^i(0) = \mathsf{u}^i(0),$$
$$\mathsf{u}^i(t) = \mathsf{u}^i\mathrm{h}(t), \quad i \in \{1,\dots,N\},$$

in which the reference value of the agents enters the dynamics as an external input, results in the same frequency representation (10a) [here, $\mathrm{h}(t)$ is the Heaviside step function]. Therefore, convergence to the average of reference values is guaranteed. Based on this observation, [49] proposes one of the earliest algorithms for dynamic average consensus:

$$\dot{x}^i(t) = -\sum_{j=1}^{N} a_{ij}(x^i(t) - x^j(t)) + \dot{\mathsf{u}}^i(t), \quad i \in \{1,\dots,N\}, \tag{11a}$$

$$x^i(0) = \mathsf{u}^i(0). \tag{11b}$$

Using a Laplace-domain analysis, [49] shows that, if each input signal $\mathsf{u}^i, i \in \{1,\dots,N\}$, has a Laplace transform with all poles in the left-half plane and at most one zero pole (such signals are asymptotically constant), then all of the agents implementing (11) over a connected graph track $\mathsf{u}^{\text{avg}}(t)$ with zero error asymptotically. As shown later, the convergence properties of (11) can be described more comprehensively using time-domain ISS analysis.

Define the tracking error of agent i by

$$e^i(t) = x^i(t) - \mathsf{u}^{\text{avg}}(t), \quad i \in \{1,\dots,N\}.$$

To analyze the system, the error is decomposed into the *consensus direction* (the direction $\mathbf{1}_N$) and the *disagreement directions* (the directions orthogonal to $\mathbf{1}_N$). To this end, define the transformation matrix $\mathbf{T} = [(1/\sqrt{N})1_N\ \mathbf{R}]$ where $\mathbf{R} \in \mathbb{R}^{N\times(N-1)}$ is such that $\mathbf{T}^\top\mathbf{T} = \mathbf{T}\mathbf{T}^\top = \mathbf{I}_N$, and consider the change of variables

$$\bar{\mathbf{e}} = \begin{bmatrix} \bar{e}_1 \\ \bar{\mathbf{e}}_{2:N} \end{bmatrix} = \mathbf{T}^\top\mathbf{e}. \tag{12}$$

In the new coordinates, (11) takes the form

$$\dot{\bar{e}}_1 = 0, \quad \bar{e}_1(t_0) = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}(x^j(t_0) - \mathsf{u}^j(t_0)), \tag{13a}$$

$$\dot{\bar{\mathbf{e}}}_{2:N} = -\mathbf{R}^\top\mathbf{L}\mathbf{R}\bar{\mathbf{e}}_{2:N} + \mathbf{R}^\top\dot{\mathbf{u}}, \quad \bar{\mathbf{e}}_{2:N}(t_0) = \mathbf{R}^\top\mathbf{x}(t_0), \tag{13b}$$

where t_0 is the initial time. Using the ISS bound on the trajectories of LTI systems (see "Input-to-State Stability of Linear Time-Invariant Systems"), the tracking error of

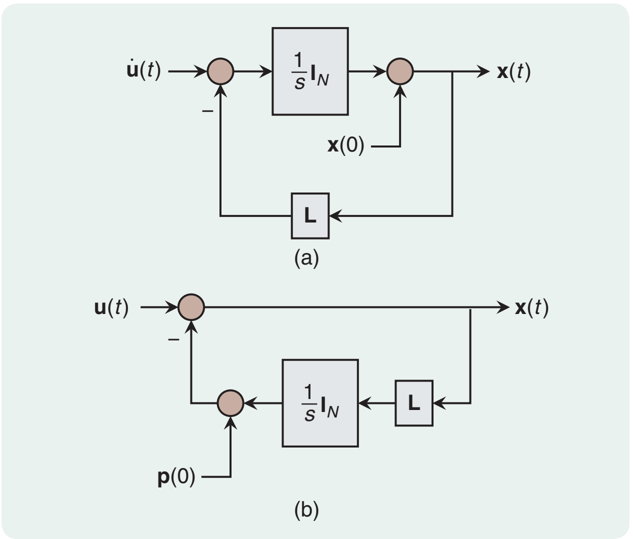


FIGURE 7 A block diagram of the continuous-time dynamic average consensus algorithms (11) and (16). Whereas the reference signals are applied as initial conditions for the static consensus algorithms, the reference signals are applied here as inputs to the system. Although both systems are equivalent, system (a) is in the form (3) and explicitly requires the derivative of the reference signals, and system (b) does not require differentiating the reference signals. (a) Dynamic average consensus algorithm (11). (b) Dynamic average consensus algorithm (16).

each agent $i \in \{1, \ldots, N\}$ while implementing (11) over a strongly connected and weight-balanced digraph is seen in (14), shown at the bottom of the page, for all $t \in [t_0, \infty)$, where $\hat{\lambda}_2$ is the smallest nonzero eigenvalue of $\mathrm{Sym}(\mathsf{L})$. [Here, $\mathbf{\Pi} = (\mathbf{I}_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)$ is used.]

The tracking error bound (14) reveals several interesting facts. First, it highlights the necessity for the special initialization $\Sigma_{j=1}^N x^j(t_0) = \Sigma_{j=1}^N \mathsf{u}^j(t_0)$ [(11b) satisfies this initialization condition]. Without it, a fixed offset from perfect tracking is present regardless of the type of reference input signals. Instead, it is expected that a proper dynamic consensus algorithm should be capable of perfectly tracking static reference signals. Next, (14) shows that (11) renders perfect asymptotic tracking not only for reference input signals with decaying rate but also for unbounded reference signals whose uncommon parts asymptotically converge to a constant value. This is due to the ISS tracking bound depending on $\|(\mathbf{I}_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}(\tau)\|$ rather than on $\|\dot{\mathbf{u}}(\tau)\|$. Note that if the reference signal of each agent $i \in \{1, \ldots, N\}$ can be written as $\mathsf{u}^i(t) = \underline{\mathsf{u}}(t) + \hat{\mathsf{u}}^i(t)$, where $\underline{\mathsf{u}}(t)$ is the (possibly unbounded) common part and $\hat{\mathsf{u}}^i(t)$ is the uncommon part of the reference signal, then

$$\left\|\left(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top\right)\dot{\mathsf{u}}(\tau)\right\| = \left\|\left(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top\right)(\dot{\underline{\mathsf{u}}}(t)\mathbf{1}_N + \dot{\hat{\mathsf{u}}}(t))\right\| = \left\|\left(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top\right)\dot{\hat{\mathsf{u}}}(t)\right\|.$$

This demonstrates that (11) properly uses the local knowledge of the unbounded but common part of the reference dynamic signals to compensate for the tracking error that would be induced due to the natural lag in diffusion of information across the network for dynamic signals. Finally, the tracking error bound (14) shows that as long as the uncommon part of the reference signals has a bounded rate, then (11) tracks the average with some bounded error. For convenience, the convergence guarantees of (11) are summarized in the following result.

Theorem 2: Convergence of (11) Over a Strongly Connected and Weight-Balanced Digraph

Let $\mathcal{G}$ be a strongly connected and weight-balanced digraph. Let

$$\sup_{\tau \in [t,\infty)} \left\|\left(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top\right)\dot{\mathbf{u}}(\tau)\right\| = \gamma(t) < \infty.$$

The trajectories of (11) are then bounded and satisfy

$$\lim_{t\to\infty} |x^i(t) - \mathsf{u}^{\mathrm{avg}}(t)| \le \frac{\gamma(\infty)}{\hat{\lambda}_2}, \quad i \in \{1, \ldots, N\}, \tag{15}$$

provided $\Sigma_{j=1}^N x^j(t_0) = \Sigma_{j=1}^N \mathsf{u}^j(t_0)$. The convergence rate to this error bound is no worse than $\mathrm{Re}(\lambda_2)$. Moreover, $\Sigma_{j=1}^N x^j(t) = \Sigma_{j=1}^N \mathsf{u}^j(t)$ for $t \in [t_0, \infty)$. ☐

The explicit expression (15) for the tracking error performance is of value for designers. The smallest nonzero eigenvalue $\hat{\lambda}_2$ of the symmetric part of the graph Laplacian is a measure of connectivity of a graph [50], [51]. For highly connected graphs (those with large $\hat{\lambda}_2$), it is expected that the diffusion of information across the graph is faster. Therefore, the tracking performance of a dynamic average consensus algorithm over such graphs should be better. Alternatively, when the graph connectivity is low, the opposite effect is expected. The ultimate tracking bound (15) highlights this inverse relationship between graph connectivity and steady-state tracking error. Given this inverse relationship, a designer can decide on the communication range of the agents and the expected tracking performance. Various upper bounds of $\hat{\lambda}_2$ that are a function of other graph invariants (such as graph degree or the network size for special families of the graphs [51], [52]) can be exploited to design the agents' interaction topology and yield an acceptable tracking performance.

Implementation Challenges and Solutions

Next, some of the features of (11) are discussed from an implementation perspective. First, note that although (11) tracks $\mathsf{u}^{\mathrm{avg}}(t)$ with a steady-state error (15), the error can be made infinitesimally small by introducing a high gain $\beta \in \mathbb{R}_{>0}$ to write L as $\beta\mathsf{L}$. By doing this, the tracking error becomes $\lim_{t\to\infty}|x^i(t) - \mathsf{u}^{\mathrm{avg}}(t)| \le (\gamma(\infty)/\beta\hat{\lambda}_2)$, $i \in \{1, \ldots, N\}$. However, for scenarios where the agents are first-order physical systems $\dot{x}^i = c^i(t)$, the introduction of this high gain results in an increase of the control effort $c^i(t)$. To address this, a balance between the control effort and tracking error margin can be achieved by introducing a two-stage algorithm in which an internal dynamics creates the average using a high-gain dynamic consensus algorithm and feeds the agreement state of the dynamic consensus algorithm as a reference signal to the physical dynamics. This approach is discussed further in the section "Controlling the Rate of Convergence."

A concern that may exists with (11) is that it requires explicit knowledge of the derivative of the reference signals. In applications where the input signals are measured online, computing the derivative can be costly and prone to

$$|e^i(t)| \le \sqrt{\|\bar{\mathrm{e}}_{2:N}(t)\|^2 + |\bar{e}_1(t)|^2} \le \sqrt{\left(\mathrm{e}^{-\hat{\lambda}_2(t-t_0)}\|\mathbf{\Pi x}(t_0)\| + \frac{\sup_{t_0\le\tau\le t}\|\mathbf{\Pi}\dot{\mathbf{u}}(\tau)\|}{\hat{\lambda}_2}\right)^2 + \left(\frac{\sum_{j=1}^N (x^j(t_0) - \mathsf{u}^j(t_0))}{\sqrt{N}}\right)^2}, \tag{14}$$

error. The other concern is the particular initialization condition requiring $\Sigma_{i=1}^{N} x^i(t_0) = \Sigma_{i=1}^{N} \mathsf{u}^i(t_0)$. To comply with this condition in a distributed setting, agents must initialize with $x^i(t_0) = \mathsf{u}^i(t_0)$. If the agents are acquiring their signal u^i from measurements or the signal is the output of a local process, any perturbation in $\mathsf{u}^i(t_0)$ results in a steady-state error in the tracking process. Moreover, if an agent (agent N) leaves the operation permanently at any time $\bar{t}$, then $\Sigma_{i=1}^{N-1} x^i$ is no longer equal to $\Sigma_{i=1}^{N-1} \mathsf{u}^i$ after $\bar{t}$. Therefore, the remaining agents (without reinitialization) carry over a steady-state error in their tracking signal.

Interestingly, all of these concerns except for the one regarding an agent's permanent departure can be resolved by a change of variables, corresponding to an alternative implementation of (11). Let $p^i = \mathsf{u}^i - x^i$ for $i \in \{1, \ldots, N\}$. Equation (11) may then be written in the equivalent form

$$\dot{p}^i(t) = \sum_{j=1}^{N} a_{ij}(x^i(t) - x^j(t)), \quad \sum_{j=1}^{N} p^j(t_0) = 0, \quad i \in \{1, \ldots, N\}, \tag{16a}$$

$$x^i(t) = \mathsf{u}^i(t) - p^i(t). \tag{16b}$$

Doing so eliminates the need to know the derivative of the reference signals and generates the same trajectories $t \mapsto x^i(t)$ as (11). We note that the initialization condition $\Sigma_{i=1}^{N} p^i(t_0) = 0$ can be easily satisfied if each agent $i \in \{1, \ldots, N\}$ starts at $p^i(0) = 0$. Note that this requirement is mild because p^i is an internal state for agent i and, therefore, is not affected by communication errors. This initialization condition, however, limits the use of (16) in applications where agents join or permanently leave the network at different points in time. To demonstrate the robustness of (16) to measurement disturbances, note that any bounded perturbation in the reference input does not affect the initialization condition but, rather, appears as an additive disturbance in the communication channel. In particular, observe the following:

$$\begin{aligned}(11\text{a}) &\Rightarrow \sum_{i=1}^{N} \dot{x}^i(t) = \sum_{i}^{N} \dot{\mathsf{u}}^i(t) \\ &\Rightarrow \sum_{i=1}^{N} x^i(t) = \sum_{i}^{N} \mathsf{u}^i(t) + \left(\sum_{i=1}^{N} x^i(t_0) - \sum_{i}^{N} \mathsf{u}^i(t_0)\right),\end{aligned} \tag{17a}$$

$$(16\text{a}) \Rightarrow \sum_{i=1}^{N} \dot{p}^i(t) = 0 \Rightarrow \sum_{i=1}^{N} p^i(t) = \sum_{i=1}^{N} p^i(t_0). \tag{17b}$$

As seen in (17a), if $\Sigma_{i=1}^{N} x^i(t_0) \neq \Sigma_{i}^{N} \mathsf{u}^i(t_0)$, then $\Sigma_{i=1}^{N} x^i(t) \neq \Sigma_{i}^{N} \mathsf{u}^i(t)$ persists in time. Therefore, if the perturbation on the reference input measurement is removed, then (11) still inherits the adverse effect of the initialization error. Instead, as (17b) shows for the case of the alternative algorithm (16), $\Sigma_{i=1}^{N} p^i(t) = 0$ is preserved in time as long as the algorithm is initialized such that $\Sigma_{i=1}^{N} p^i(t_0) = 0$, which can be easily done by setting $p^i(t_0) = 0$ for $i \in \{1, \ldots, N\}$. Consequently, when the perturbations are removed, then (16) recovers the convergence guarantee of the perturbation-free case. Following steps similar to the ones leading to the bound (14), the effect of the additive bounded reference signal measurement perturbation on the convergence of (16) is summarized in the next result.

Lemma 1: Convergence of (16) Over a Strongly Connected and Weight-Balanced Digraph in the Presence of Additive Reference Input Perturbations

Let $\mathcal{G}$ be a strongly connected and weight-balanced digraph. Suppose $\mathsf{w}^i(t)$ is an additive perturbation on the measured reference input signal $\mathsf{u}^i(t)$. Let $\sup_{\tau \in [t,\infty)} \|(\mathbf{I}_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathsf{u}}(\tau)\| = \gamma(t) < \infty$ and $\sup_{\tau \in [t,\infty)} \|(\mathbf{I}_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{w}}(\tau)\| = \omega(t) < \infty$. Then, the trajectories of (16) are bounded and satisfy

$$\lim_{t \to \infty} |x^i(t) - \mathsf{u}^{\text{avg}}(t)| \leq \frac{\gamma(\infty) + \omega(\infty)}{\hat{\lambda}_2}, \qquad i \in \{1, \ldots, N\},$$

provided $\Sigma_{j=1}^{N} p^j(t_0) = 0$. The convergence rate to this error bound is no worse than $\text{Re}(\lambda_2)$. Moreover, $\Sigma_{j=1}^{N} p^j(t) = 0$ for $t \in [t_0, \infty)$. □

The perturbation w^i in Lemma 1 can also be regarded as a bounded communication perturbation. Therefore, (16) [and similarly (11)] is considered naturally robust to bounded communication error.

From an implementation perspective, it is also desirable that a distributed algorithm is robust to changes in the communication topology that may arise as a result of unreliable transmissions, limited communication/sensing range, network rerouting, or the presence of obstacles. To analyze this aspect, consider a time-varying digraph $\mathcal{G}(\mathcal{V}, \mathcal{E}(t), \mathbf{A}_\sigma(t))$, where the nonzero entries of the adjacency matrix $\mathbf{A}(t)$ are uniformly lower and upper bounded [in other words, $\mathsf{a}_{ij}(t) \in [\underline{a}, \bar{a}]$, where $0 < \underline{a} \leq \bar{a}$ if $(j, i) \in \mathcal{E}(t)$, and $\mathsf{a}_{ij} = 0$ otherwise]. Here, $\sigma : [0, \infty) \to \mathcal{P} = \{1, \ldots, m\}$ is a piecewise constant signal, meaning that it has only a finite number of discontinuities in any finite time interval and is constant between consecutive discontinuities. Intuitively, consensus in switching networks occurs if there is occasionally enough flow of information from every node in the network to every other node.

Formally, an admissible switching set $\mathcal{S}_{\text{admis}}$ is a set of piecewise constant switching signals $\sigma : [0, \infty) \to \mathcal{P}$ with some dwell time t_L (in other words, $t_{k+1} - t_k > t_L > 0$, for all $k = 0, 1, \ldots$), such that

- » $\mathcal{G}(\mathcal{V}, \mathcal{E}(t), \mathbf{A}_\sigma(t))$ is weight balanced for $t \geq t_0$.
- » The number of contiguous, nonempty, uniformly bounded time intervals $[t_{i_j}, t_{i_{j+1}})$, $j = 1, 2, \ldots$, starting at $t_{i_1} \geq t_0$, with the property that $\cup_{t_{i_j}}^{t_{i_{j+1}}} \mathcal{G}(\mathcal{V}, \mathcal{E}(t), \mathbf{A}_\sigma(t))$ is a jointly strongly connected digraph, goes to infinity as $t \to \infty$.

When the switching signal belongs to the admissible set $\mathcal{S}_{\text{admis}}$, [19] shows that there always exists $\underline{\lambda} \in \mathbb{R}_{>0}$ and $\kappa \in \mathbb{R}_{\geq 1}$ such that $\|\mathrm{e}^{-\mathbf{R}^\top \mathbf{L}_\sigma \mathbf{R}}\| \leq \kappa \mathrm{e}^{-\underline{\lambda} t}$, $t \in \mathbb{R}_{\geq 0}$. Implementing the change of variables (12), it is shown that the trajectories of (16) satisfy (14), with $\hat{\lambda}_2$ replaced by $\underline{\lambda}$ and $\|\mathbf{\Pi}\mathbf{x}(t_0)\|$ and $\|\mathbf{\Pi}\dot{\mathbf{u}}(\tau)\|$ being multiplied by κ. This statement is formalized as follows.

Lemma 2: Convergence of (11) Over Switching Graphs

Let the communication topology be $\mathcal{G}(\mathcal{V},\mathcal{E}(t),\mathbf{A}_\sigma(t))$ where $\sigma \in \mathcal{S}_{\text{admis}}$. Let $\sup_{\tau\in[t,\infty)}\|(\mathbf{I}_N-(1/N)\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}(\tau)\| = \gamma(t) < \infty$. The trajectories of (11) are bounded and satisfy

$$\lim_{t\to\infty}|x^i(t)-\mathsf{u}^{\text{avg}}(t)| \le \frac{\kappa\gamma(\infty)}{\underline{\lambda}}, \qquad i\in\{1,\dots,N\},$$

provided $\Sigma_{j=1}^N x^j(t_0) = \Sigma_{j=1}^N \mathsf{u}^j(t_0)$. The convergence rate to this error bound is no worse than $\underline{\lambda}$. Moreover, we have $\Sigma_{j=1}^N x^j(t) = \Sigma_{j=1}^N \mathsf{u}^j(t)$ for $t\in[t_0,\infty)$. □

Example: Distributed Formation Control Revisited

We revisit one of the scenarios discussed in the "Applications of Dynamic Average Consensus in Network Systems" section to illustrate the properties of (11) and its alternative implementation (16). Consider a group of four mobile agents (depicted as the triangle robots in Figure 8) whose communication topology is described by a fixed, connected, undirected ring. The objective of these agents is to follow a set of moving targets (depicted as the round robots in Figure 8) in a containment fashion (that is, ensuring that they are surrounded as they move around the environment). Let

$$x_T^l(t) = (t/20)^2 + 0.5\sin\left((0.35+0.05l)\,t + (5-l)\frac{\pi}{5}\right) + 4 - 2(l-1), \quad l\in\{1,2,3,4\} \tag{18}$$

be the horizontal position of the set of moving targets (each mobile agent tracks one moving target). The term $(t/20)^2$ in the reference signals (18) represents the component with an unbounded derivative that is common to all agents.

To achieve their objective, the group of agents seeks to compute on the fly the geometric center $\bar{x}_T(t) = (1/N)\Sigma_{l=1}^N x_T^l(t)$ and the associated variance $(1/N)\Sigma_{l=1}^N (x^l(t)-\bar{x}_T(t))^2$ determined by the time-varying position of the moving targets. The agents implement two distributed dynamic average consensus algorithms: one for computing the center and the other for computing the variance (as shown in Figure 9). To illustrate the properties discussed in this section, consider that agent 4 (the green triangle in Figure 8) leaves the network 10 s after the

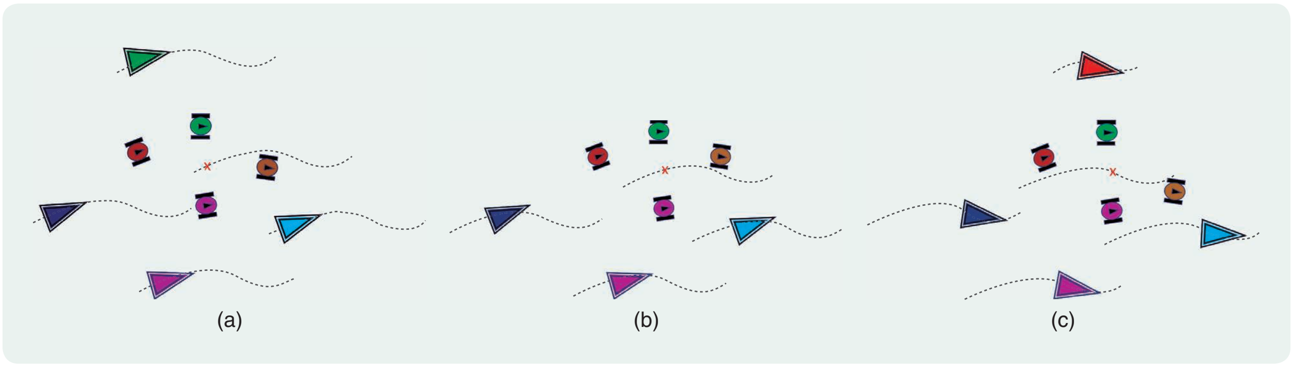


FIGURE 8 A simple dynamic average consensus-based containment and tracking of a team of mobile targets. (a) The triangle robots cooperatively want to contain the moving round robots by making a formation around the geometric center of the round robots that they are observing. At (b) (after, for example, 10 s from the start of the operation), one of the triangle robots leaves the team. At (c) (after, for example, 20 s from the start of the operation), a new triangle robot joins the group to take over tracking the abandoned round robot.

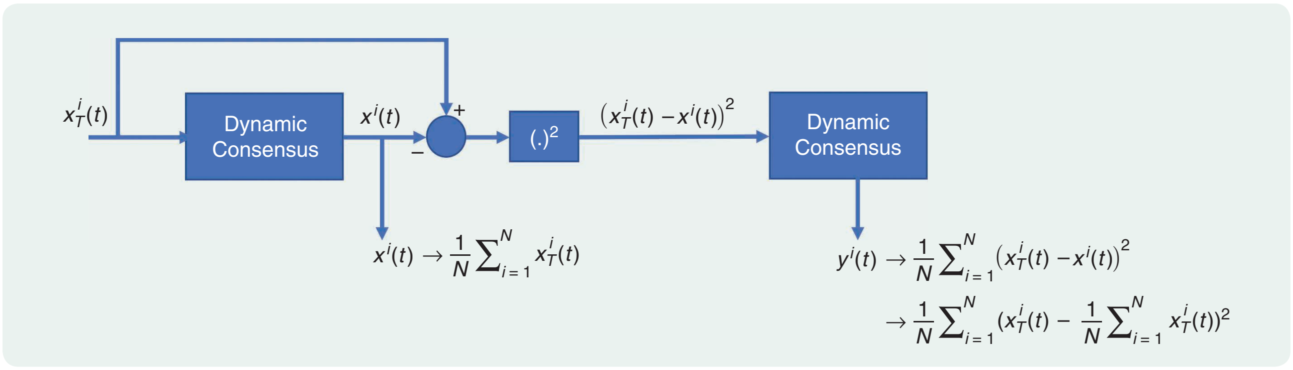

$x_T^i(t)$
Dynamic Consensus
$x^i(t)$
+
–
$(.)^2$
$(x_T^i(t)-x^i(t))^2$
Dynamic Consensus
$x^i(t) \to \frac{1}{N}\sum_{i=1}^N x_T^i(t)$
$y^i(t) \to \frac{1}{N}\sum_{i=1}^N (x_T^i(t)-x^i(t))^2$
$\to \frac{1}{N}\sum_{i=1}^N (x_T^i(t) - \frac{1}{N}\sum_{i=1}^N x_T^i(t))^2$

FIGURE 9 A group of N mobile agents uses a set of dynamic consensus algorithms to asymptotically track the geometric center $x_T^{\text{avg}}(t) = (1/N)\sum_{i=1}^N x_T^i(t)$ and its variance $(1/N)\sum_{i=1}^N (x^i(t)-x_T^{\text{avg}}(t))^2$. In this setup, each mobile agent $i\in\{1,\dots,N\}$ is monitoring its respective target's position $x_T^i(t)$.

beginning of the simulation. After 10 s, a new agent, labeled 5 (the red triangle in Figure 8), joins the network and starts monitoring the target that agent 4 was in charge of. For simplicity, the simulation is focused on the calculation of the geometric center. For this computation, agents implement (16) with reference input $\mathsf{u}^i(t) = x_T^i(t),\ i \in \{1,2,3,4\}$. Figure 10 shows the algorithm performance for various operational scenarios. As forecast by the discussion of this section, the tracking error vanishes in the presence of perturbations in the input signals available to the individual agents and switching topologies, and it exhibits only partial robustness to agent arrivals, departures, and initialization errors, with a constant bias with respect to the correct average. Additionally, it is worth noticing in Figure 10 that all of the agents exhibit convergence with the same rate.

The introduction of (16) serves as preparation for a more in-depth treatment of the design of dynamic average consensus algorithms. This includes a discussion of the issues of correct initialization [the steady-state error depends on the initial condition $\mathbf{x}(0)$ or $\mathbf{x}_0$], adjusting the convergence rate of the agents, and the limitation of tracking (with zero steady-state error) only constant reference signals (and therefore with small steady-state error for slowly time-varying reference signals). To improve clarity, continuous-time and discrete-time strategies are discussed separately.

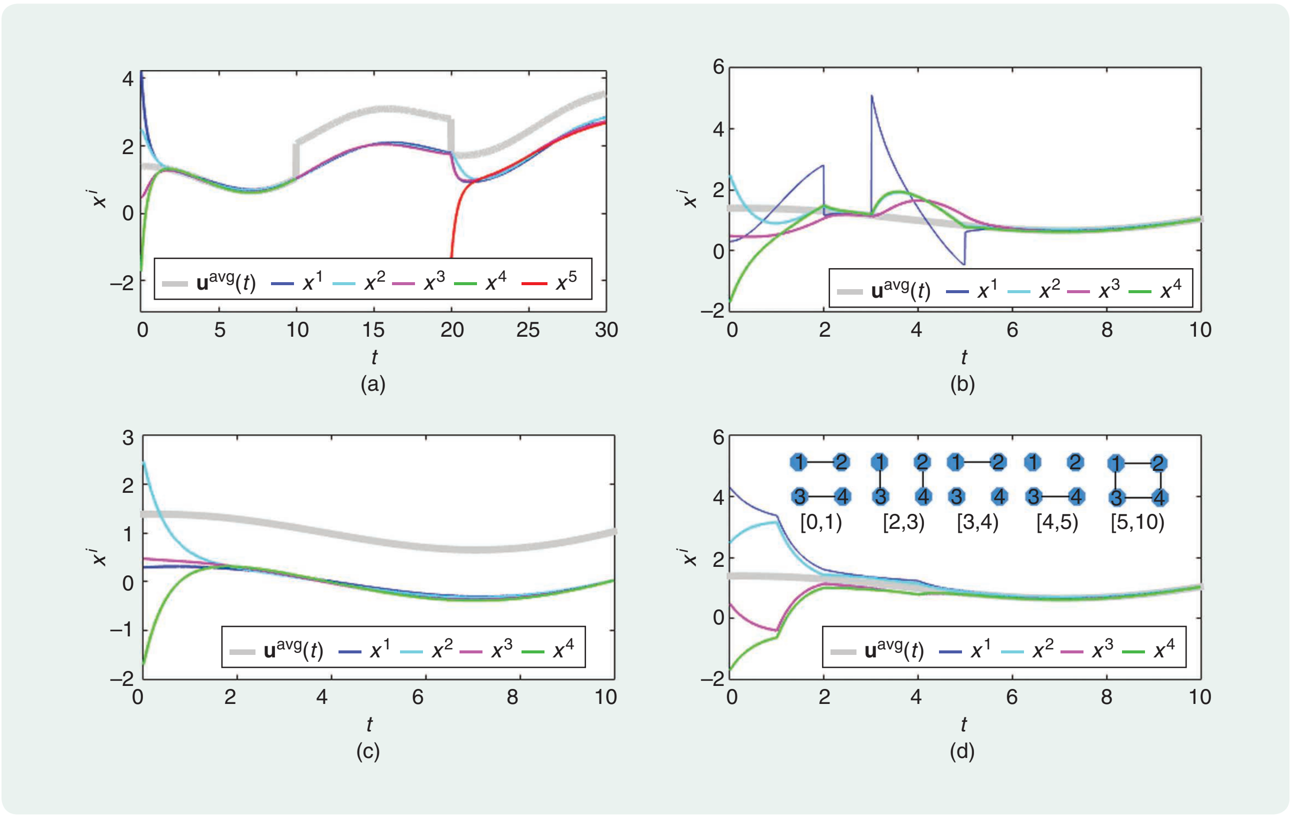


FIGURE 10 A performance evaluation of dynamic average consensus algorithms (11) and (16) for a group of four agents with reference inputs (18) for a tracking scenario described in Figure 8. In the simulation in plot (a), the agents are using (16). As guaranteed in Lemma 1 [under proper initialization $p^i(0) = 0, i \in \{1,2,3,4\}$] during the time interval [0,10] s, the agents are able to track $x_T^{\text{avg}}(t)$ with a small error. The challenge presents itself when agent 4 leaves the operation at $t = 10$ s. Because after agent 4 leaves $\Sigma_{i=1}^{3} p^i(10^+) \neq 0$, the remaining agents fail to follow the average of their reference values, which now is $(1/3)\Sigma_{l=1}^{3} \mathsf{u}^l$. Similarly, even with initialization of $p^5(20) = 0$ for the new agent 5, because $p^1(20) + p^2(20) + p^3(20) + p^5(20) \neq 0$, the agents track the average $(1/4)\Sigma_{l=1}^{4} x_T^l(t)$ with a steady-state error. In the simulation in plot (b), the agents are using (16), and at time interval [0, 10] s, agent 1's reference input is subject to a measurement perturbation according to $\mathsf{u}^1(t) = x_T^1(t) + \mathsf{w}^1(t)$, where $\mathsf{w}^1(t) = -4\cos(t)$ at $t \in [0, 2]$, and $t \in [3, 5]$ and $\mathsf{w}^1(t) = 0$ at other times. As guaranteed by Lemma 1, despite the perturbation, including the initial measurement error of $\mathsf{u}^1(0) = x_T^1(0) - 4$, (16) has robustness to the measurement perturbation and recovers its performance after the perturbation is removed. A large perturbation error was used, so that its effect is observed more visibly in the simulation plots. In the simulation in plot (c), the agents are using (11). Agent 1's reference input has an initial measurement error of $\mathsf{u}^1(0) = x_T^1(0) - 4$. Because the measurement error directly affects the initialization condition of the algorithm, it fails to preserve $\Sigma_{l=1}^{4} x^l(t) = \Sigma_{l=1}^{4} u^l(t)$. As a result, the effect of initialization error persists, and the algorithm maintains a significant tracking error. In the simulation in plot (d), the agents are using (16) [similar results are also obtained for (11)]. The network communication topology is a switching graph, where the graph topology at different time intervals is shown on the plot. Because the switching signal σ belongs to $\mathcal{S}_{\text{admis}}$, as predicted by Lemma 2, the trajectories of the algorithm stay bounded, and once the topology becomes fully connected, the agents follow their respective dynamic average closely. (a) Agent departure and arrival. (b) Perturbation of input signals. (c) Initialization error. (d) Switching topology.

CONTINUOUS-TIME DYNAMIC AVERAGE CONSENSUS ALGORITHMS

This section discusses various continuous-time dynamic average consensus algorithms and their performance and robustness guarantees. Table 1 summarizes the arguments of the driving command of these algorithms in (1) and their special initialization requirements. Some of these algorithms, when cast in the form of (1), require access to the derivative of the reference signals. Similar to (11), however, this requirement can be eliminated using alternative implementations.

Robustness to Initialization and Permanent Agent Dropout

To eliminate the special initialization requirement and induce robustness with respect to algorithm initialization, [53] proposes the following alternative dynamic average consensus algorithm:

$$\dot{q}^i(t) = -\sum_{j=1}^{N} b_{ij}(x^i - x^j), \tag{19a}$$

$$\dot{x}^i = -\alpha(x^i - u^i) - \sum_{j=1}^{N} a_{ij}(x^i - x^j) + \sum_{j=1}^{N} b_{ji}(q^i - q^j) + \dot{\mathrm{u}}^i, \tag{19b}$$

$$q^i(t_0),\, x^i(t_0) \in \mathbb{R}, \qquad i \in \{1,\dots,N\}, \tag{19c}$$

where $\alpha \in \mathbb{R}_{>0}$. Here, $\dot{\mathrm{u}}^i$ is added to (19b) to allow agents to track reference inputs whose derivatives have unbounded common components. The necessity of having explicit knowledge of the derivative of reference signals can be removed by using the change of variables $p^i = x^i - \mathrm{u}^i$, $i \in \{1,\dots,N\}$. In (19), the agents are allowed to use two different adjacency matrices, $[a_{ij}]_{N\times N}$ and $[b_{ij}]_{N\times N}$, so that they have an extra degree of freedom to adjust the tracking performance of the algorithm. The Laplacian matrices associated with adjacency matrices $[a_{ij}]$ and $[b_{ij}]$ are represented by, respectively, $\mathbf{L}_\mathrm{P}$ (labeled as proportional Laplacian) and $\mathbf{L}_\mathrm{I}$ (labeled as integral Laplacian). The compact representation of (19) is

$$\dot{\mathbf{q}} = -\mathbf{L}_\mathrm{I}\mathbf{x}, \tag{20a}$$

$$\dot{\mathbf{x}} = -\alpha(\mathbf{x} - \mathbf{u}) - \mathbf{L}_\mathrm{p}\mathbf{x} + \mathbf{L}_\mathrm{I}^\top \mathrm{q} + \dot{\mathbf{u}}, \tag{20b}$$

which also reads as

$$\dot{\mathbf{x}} = -\alpha(x - \mathbf{u}) - \mathbf{L}_\mathrm{p}\mathbf{x} - \mathbf{L}_\mathrm{I}^\top \int_{t_0}^{t} \mathbf{L}_\mathrm{I}\mathbf{x}(\tau)\,\mathrm{d}\tau + \mathbf{L}_\mathrm{I}^\top \mathbf{q}(t_0) + \dot{\mathbf{u}}.$$

Using a time-domain analysis similar to that employed for (11), the ultimate tracking behavior of (19) is characterized. Consider the change of variables (12) and

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \mathbf{w}_{2:N} \end{bmatrix} = \mathbf{T}^\top \mathbf{q}, \tag{21a}$$

$$\mathbf{y} = \mathbf{w}_{2:N} - \alpha(\mathbf{R}^\top \mathbf{L}_\mathrm{I}^\top \mathbf{R})^{-1}\mathbf{R}^\top \dot{\mathbf{u}}, \tag{21b}$$

to write (20) in the equivalent form

$$\dot{w}_1 = 0, \tag{22a}$$

$$\begin{bmatrix} \dot{\mathbf{y}} \\ \dot{\bar{e}}_1 \\ \dot{\bar{\mathbf{e}}}_{2:N} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{R}^\top \mathbf{L}_\mathrm{I} \mathbf{R} \\ 0 & -\alpha & 0 \\ \mathbf{R}^\top \mathbf{L}_\mathrm{I}^\top \mathbf{R} & 0 & -\alpha \mathbf{I} - \mathbf{R}^\top \mathbf{L}_\mathrm{p} \mathbf{R} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \mathbf{y} \\ \bar{e}_1 \\ \bar{\mathbf{e}}_{2:N} \end{bmatrix} + \underbrace{\begin{bmatrix} -\alpha(\mathbf{R}^\top \mathbf{L}_\mathrm{I}^\top \mathbf{R})^{-1} \\ 0 \\ \mathbf{I} \end{bmatrix}}_{\mathbf{B}} \mathbf{R}^\top \dot{\mathbf{u}}. \tag{22b}$$

Let the communication ranges of the agents be such that they can establish adjacency relations $[a_{ij}]$ and $[b_{ij}]$ so that the corresponding $\mathbf{L}_\mathrm{I}$ and $\mathbf{L}_\mathrm{P}$ are Laplacian matrices of strongly connected and weight-balanced digraphs. Invoking [53, Lemma 9], matrix $\mathbf{A}$ in (22b) is shown to be Hurwitz. Therefore, using the ISS bound on the trajectories of LTI systems (see "Input-to-State Stability of Linear Time-Invariant Systems"), the tracking error of each agent $i \in \{1,\dots,N\}$ while implementing (19) over a strongly connected and weight-balanced digraph is

$$|e^i(t)| \le \kappa e^{-\underline{\lambda}(t-t_0)} \left\| \begin{bmatrix} \mathbf{w}_{2:N}(t_0) \\ \bar{\mathbf{e}}(t_0) \end{bmatrix} \right\| + \frac{\kappa \|\mathbf{B}\|}{\underline{\lambda}} \sup_{t_0 \le \tau \le t} \left\| \left(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N \mathbf{1}_N^\top\right) \dot{\mathbf{u}}(\tau) \right\|, \tag{23}$$

where $(\kappa, \underline{\lambda})$ are given by (S5) for matrix $\mathbf{A}$ of (22b) and can be computed from (S9). It is shown that both κ and $\underline{\lambda}$ depend on the smallest nonzero eigenvalues of $\mathrm{Sym}(\mathbf{L}_\mathrm{I})$ and $\mathrm{Sym}(\mathbf{L}_\mathrm{p})$ as well as α. Therefore, the tracking performance of (19) depends on both the magnitude of the derivative of reference signals and the connectivity of the communication graph. From this error bound, it is observed that, for bounded dynamic signals with bounded rate, (19) is guaranteed to track the dynamic average with an ultimately bounded error. Moreover, this algorithm does not need any special initialization. The robustness to initialization can be observed on the block diagram representation of (19),

TABLE 1 The arguments of the driving command in (1) for the reviewed continuous-time dynamic average consensus algorithms together with their initialization requirements.

Algorithm	(11)	(19)	(24)	(25)
$J^i(t)$	$\{x^i(t), \dot{\mathrm{u}}(t)\}$	$\{x^i(t), v^i(t), \mathrm{u}(t)\}$	$\{x^i(t), z^i(t), v^i(t), \mathrm{u}(t), \dot{\mathrm{u}}(t)\}$	$\{x^i(t), v^i(t), \mathrm{u}(t), \dot{\mathrm{u}}(t)\}$
$\{I^j(t)\}_{j\in N^i_{\mathrm{out}}}$	$\{x^j(t)\}_{j\in N^i_{\mathrm{out}}}$	$\{x^j(t), v^j(t)\}_{j\in N^i_{\mathrm{out}}}$	$\{z^j(t), v^j(t)\}_{j\in N^i_{\mathrm{out}}}$	$\{v^j(t)\}_{j\in N^i_{\mathrm{out}}}$
Initialization requirement	$x^i(0) = \mathrm{u}^i(0)$	None	None	$\sum_{j=1}^{N} v^j(0) = 0$

shown in Figure 11(a). For reference, the convergence guarantees of algorithm (19) are summarized next.

Theorem 3: Convergence of (19)

Let $\mathbf{L}_P$ and $\mathbf{L}_I$ be Laplacian matrices corresponding to strongly connected and weight-balanced digraphs. Let $\sup_{\tau \in [t,\infty)} \|(\mathbf{I}_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}(\tau)\| = \gamma(t) < \infty$. Starting from any $x^i(t_0), q(t_0) \in \mathbb{R}$, for any $\alpha \in \mathbb{R}_{>0}$ the trajectories of algorithm (25) satisfy

$$\lim_{t\to\infty} |x^i(t) - \mathsf{u}^{\text{avg}}(t)| \le \frac{\kappa \|\mathbf{B}\| \gamma(\infty)}{\underline{\lambda}}, \quad i \in \{1,\dots,N\},$$

where $\kappa, \underline{\lambda} \in \mathbb{R}_{>0}$ satisfy $\|\mathbf{e}^{\mathbf{A}t}\| \le \kappa \mathbf{e}^{-\underline{\lambda} t}$ [$\mathbf{A}$ and $\mathbf{B}$ are given in (22b)]. Moreover,

$$\sum_{j=1}^{N} x^j(t) = \sum_{j=1}^{N} \mathsf{u}^j(t) + \mathbf{e}^{-\alpha(t-t_0)}\left(\sum_{j=1}^{N} x^j(t_0) - \sum_{j=1}^{N} \mathsf{u}^j(t_0)\right),$$

for $t \in [t_0, \infty)$. □

Figure 12 shows the performance of (19) in the distributed formation control scenario represented in Figure 8. This plot illustrates how the property of robustness to the initialization error of (19) allows it to accommodate the addition and deletion of agents with satisfactory tracking performance.

Although the convergence guarantees of (19) are valid for strongly connected and weight-balanced digraphs, from an implementation perspective, the use of this strategy over directed graphs may not be feasible. In fact, the presence of the transposed integral Laplacian $\mathbf{L}_I^\top$ in (20b) requires each agent $i \in \{1,\dots,N\}$ to know not only the entries in row i but also the column i of $\mathbf{L}_I$ and receive information from the corresponding agents. However, for undirected graph topologies, this requirement is satisfied trivially as $\mathbf{L}_I^\top = \mathbf{L}_I$.

Controlling the Rate of Convergence

A common feature of the dynamic average consensus algorithms presented in the "A First Design for Dynamic Average Consensus" and "Robustness to Initialization and Permanent Agent Dropout" sections is that the rate of convergence is the same for all agents and dictated by network topology as well as some algorithm parameters [see (14) and (23)]. However, in some applications, the task is not just to obtain the average of the dynamic inputs but rather to physically track this value, possibly with limited control authority. To allow the network to prespecify its desired worst rate of convergence β, [54] proposes dynamic average consensus algorithms whose design incorporates two time scales. The *first-order-input dynamic consensus* (*FOI-DC*) algorithm is described as

$$\begin{cases} \epsilon \dot{q}^i = -\sum_{j=1}^{N} b_{ij}(z^i - z^j), \\ \epsilon \dot{z}^i = -(z^i + \beta \mathsf{u}^i + \dot{\mathsf{u}}^i) - \sum_{j=1}^{N} a_{ij}(z^i - z^j) + \sum_{j=1}^{N} b_{ji}(q^i - q^j), \end{cases} \tag{24a}$$

$$\dot{x}^i = -\beta x^i - z^i, \quad i \in \{1,\dots,N\}. \tag{24b}$$

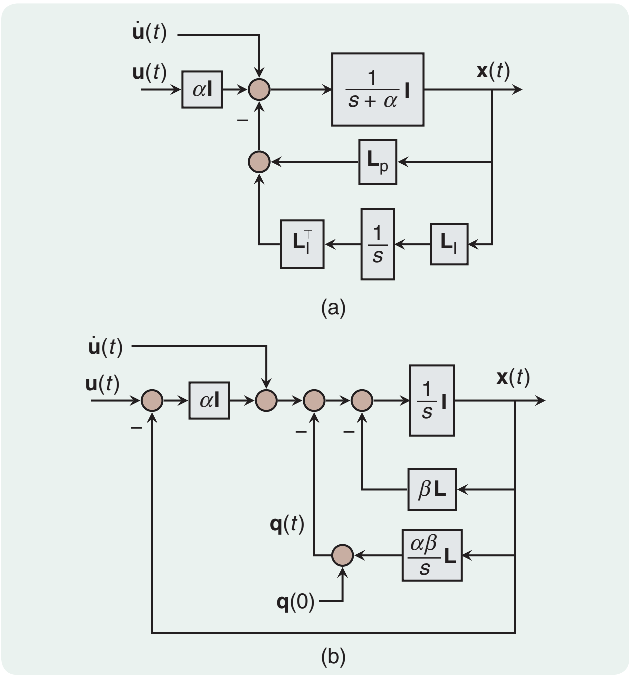


FIGURE 11 A block diagram of continuous-time dynamic average consensus algorithms. These *dynamic* algorithms naturally adapt to changes in the reference signals, which are applied as inputs to the system. Continuous-time algorithm (19) is robust to initialization. To see why the algorithm is robust, consider multiplying the input signal on the left in plot (a) by $1_N^\top$. The output of the integrator block $(1/s)$ is multiplied by zero (because $\mathbf{L}_I \mathbf{1}_N = 0$) and therefore does not affect the output. Although the output is affected by the initial state of the $1/(s+\alpha)$ block, this term decays to zero and therefore does not affect the steady state. Also, the requirement of needing the derivative of the input $\dot{\mathsf{u}}(t)$ can be removed by a change of variable. The continuous-time algorithm in (25) is not robust to initialization. In this algorithm, the parameter β may be used to control the tracking error size, and α may be used to control the rate of convergence. Furthermore, this algorithm is robust to reference signal measurement perturbations and naturally preserves the privacy of the input signals against adversaries [19]. (a) Continuous-time algorithm (19). (b) Continuous-time algorithm (25).

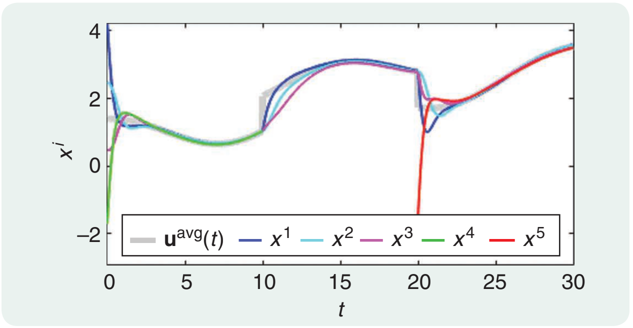


FIGURE 12 The performance of dynamic average consensus algorithm (19) in the distributed formation control scenario of Figure 8. A group of four mobile agents acquires reference inputs (18) corresponding to the time-varying position of a set of moving targets. The algorithm convergence properties are not affected by initialization errors, as stated in Theorem 3. This property also makes it robust to agent arrivals and departures. In this simulation, agent 4 leaves the network at time $t = 10$ s, and a new agent 5 joins the network at $t = 20$ s. In contrast to what was observed for (16) and (19) in Figure 10, the execution recovers its tracking performance after a transient.

The fast dynamics is (24a) and employs a small value for $\epsilon \in \mathbb{R}_{>0}$. The fast dynamics, which builds on the proportional-integral (PI) algorithm (19), is intended to generate the average of the sum of the dynamic input and its first derivative. The slow dynamics (24b) then uses the signal generated by the fast dynamics to track the average of the reference signal across the network at a prespecified smaller rate $\beta \in \mathbb{R}_{>0}$. The novelty is that these slow and fast dynamics are running simultaneously, and thus, there is no need to wait for convergence of the fast dynamics and then take slow steps toward the input average.

Similar to the dynamic average consensus algorithm (19), (24) does not require any specific initialization. The technical approach used in [54] to study the convergence of (24) is based on the singular perturbation theory [55, Ch. 11], which results in a guaranteed convergence to an ϵ-neighborhood of $\mathsf{u}^{\mathrm{avg}}(t)$ for small values of $\epsilon \in \mathbb{R}_{>0}$. Using time-domain analysis, information about the ultimate tracking behavior of (19) can be made more precise. For convenience, the changes of variables (12) and (21a) with $\mathbf{y} = \mathbf{w}_{2:N} - (\mathbf{R}^\top \mathbf{L}_{\mathrm{I}}^\top \mathbf{R})^{-1}\mathbf{R}^\top(\beta\mathbf{u} + \dot{\mathbf{u}})$ and $\mathbf{e}_z = \mathbf{T}^\top(\mathbf{z} + (\beta/N)\Sigma_{j=1}^{N}\mathsf{u}^j\mathbf{1}_N + (1/N)\Sigma_{j=1}^{N}\dot{\mathsf{u}}^j\mathbf{1}_N)$ are applied to write the FOI-DC algorithm as

$$\dot{w}_1 = 0,$$

$$\begin{bmatrix}\dot{\mathbf{y}}\\ \dot{\mathbf{e}}_z\end{bmatrix} = \epsilon^{-1}\underbrace{\begin{bmatrix}\mathbf{0} & [\mathbf{0} \;\; -\mathbf{R}^\top\mathbf{L}_{\mathrm{I}}\mathbf{R}]\\ \begin{bmatrix}0\\ \mathbf{R}^\top\mathbf{L}_{\mathrm{I}}^\top\mathbf{R}\end{bmatrix} & \begin{bmatrix}-1 & 0\\ \mathbf{0} & -\mathbf{I}-\mathbf{R}^\top\mathbf{L}_{\mathrm{p}}\mathbf{R}\end{bmatrix}\end{bmatrix}}_{\bar{\mathbf{A}}}\begin{bmatrix}\mathbf{y}\\ \mathbf{e}_z\end{bmatrix} + \underbrace{\begin{bmatrix}-(\mathbf{R}^\top\mathbf{L}_{\mathrm{I}}^\top\mathbf{R})^{-1}[\mathbf{0}_{(N-1)\times 1} \;\; \mathbf{I}_{N-1}]\\ \begin{bmatrix}[1 \;\; \mathbf{0}_{1\times N-1}]\\ \mathbf{0}_{(N-1)\times N}\end{bmatrix}\end{bmatrix}}_{\bar{\mathbf{B}}}\mathbf{f}(t),$$

$$\dot{\mathbf{e}} = -\beta\mathbf{e} - \mathbf{e}_z,$$

where $\mathbf{f}(t) = \mathbf{T}^\top(\beta\dot{\mathbf{u}} + \ddot{\mathbf{u}})$. Using the ISS bound on the trajectories of LTI systems (see "Input-to-State Stability of Linear Time-Invariant Systems"), the tracking error of each agent $i \in \{1,\ldots,N\}$ while implementing the FOI-DC algorithm with $\epsilon \in \mathbb{R}_{>0}$ is,

$$\begin{aligned}|e^i(t)| \le\; & \mathbf{e}^{-\beta(t-t_0)}|e^i(t_0)|\\ & + \frac{\kappa}{\beta}\sup_{t_0\le\tau\le t}\left(\mathbf{e}^{-\epsilon^{-1}\underline{\lambda}(t-t_0)}\left\|\begin{bmatrix}\mathbf{y}(t_0)\\ \mathbf{e}_z(t_0)\end{bmatrix}\right\|\right.\\ & \left. + \frac{\epsilon\|\bar{\mathbf{B}}\|}{\underline{\lambda}}\sup_{t_0\le\tau\le t}\|\beta\dot{\mathbf{u}}(\tau) + \ddot{\mathbf{u}}(\tau)\|\right),\end{aligned}$$

where $\|\mathbf{e}^{\bar{\mathbf{A}}t}\| \le \kappa\mathbf{e}^{-\underline{\lambda}t}$. From this error bound, it is observed that, for dynamic signals with bounded first and second derivatives, the FOI-DC algorithm is guaranteed to track the dynamic average with an ultimately bounded error. This tracking error can be made small using a small $\epsilon \in \mathbb{R}_{>0}$. Use of small $\epsilon \in \mathbb{R}_{>0}$ also results in dynamics (25a) to have a higher decay rate. Therefore, the dominant rate of convergence of the FOI-DC algorithm is determined by β, which can be prespecified regardless of the interaction topology. Moreover, β can be used to regulate the control effort of the integrator dynamics $\dot{x}^i = c^i(t)$, $i \in \{1,\ldots,N\}$ while maintaining a good tracking error via the use of small $\epsilon \in \mathbb{R}_{>0}$.

An Alternative Algorithm for Directed Graphs

As observed, (19) is not implementable over directed graphs because it requires information exchange with both in- and out-neighbors, and these sets are generally different. In [19], the authors proposed a modified proportional and integral agreement feedback dynamic average consensus algorithm whose implementation does not require the agents to know their respective columns of the Laplacian. This algorithm is

$$\dot{q}^i = \alpha\beta\sum_{j=1}^{N}\mathsf{a}_{ij}(x^i - x^j), \tag{25a}$$

$$\dot{x}^i = -\alpha(x^i - \mathsf{u}^i) - \beta\sum_{j=1}^{N}\mathsf{a}_{ij}(x^i - x^j) - q^i + \dot{\mathsf{u}}^i, \tag{25b}$$

$$x^i(t_0), q^i(t_0) \in \mathbb{R} \;\text{ s.t. }\; \sum_{j=1}^{N} q^j(t_0) = 0, \tag{25c}$$

$i \in \{1,\ldots,N\}$, where $\alpha, \beta \in \mathbb{R}_{>0}$. Equation (25) in compact form can be equivalently written as

$$\dot{\mathbf{x}} = -\alpha(\mathbf{x} - \mathbf{u}) - \beta\mathbf{L}\mathbf{x} - \alpha\beta\int_{t_0}^{t}\mathbf{L}\mathbf{x}(\tau)\,d\tau - \mathbf{q}(t_0) + \dot{\mathbf{u}},$$

which demonstrates the proportional and integral agreement feedback structure of this algorithm. As was done for (11), a change of variables $p^i = \mathsf{u}^i - x^i$ can be used to write this algorithm in a form whose implementation does not require the knowledge of the derivative of the reference signals.

Note an interesting connection between (25) and (16). Writing the transfer function from the reference input to the tracking error state (25), there is a pole-zero cancellation that reduces (25) to (11) and (16). Despite this close relationship, there are some subtle differences. For example, unlike (11), (25) enjoys robustness to reference signal measurement perturbations and naturally preserves the privacy of the input of each agent against adversaries. Specifically, an adversary with access to the time history of all network communication messages cannot uniquely reconstruct the reference signal of any agent [19], which is not the case for (16).

Figure 11(b) shows the block diagram representation of this algorithm. The next result states the convergence properties of (25). See [19] for the proof of this statement, which is established using the time-domain analysis implemented to analyze the algorithms reviewed so far.

Theorem 4: Convergence of (25) Over Strongly Connected and Weight-Balanced Digraphs for Dynamic Input Signals [19]

Let $\mathcal{G}$ be a strongly connected and weight-balanced digraph. Let $\sup_{\tau\in[t,\infty)}\|(\mathbf{I}_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}(\tau)\| = \gamma(t) < \infty$. For any $\alpha, \beta \in \mathbb{R}_{>0}$, the trajectories of (25) satisfy

$$\lim_{t\to\infty}\left|x^i(t) - \mathrm{u}^{\mathrm{avg}}(t)\right| \le \frac{\gamma(\infty)}{\beta\hat{\lambda}_2}, \qquad i \in \{1,\ldots,N\}, \tag{26}$$

provided $\Sigma_{j=1}^{N} q^j(t_0) = 0$. The convergence rate to the error bound is $\min\{\alpha, \beta\,\mathrm{Re}(\lambda_2)\}$. □

The inverse relation between β and the tracking error in (26) indicates that the parameter β can be used to control the tracking error size, and α can be used to control the rate of convergence.

DISCRETE-TIME DYNAMIC AVERAGE CONSENSUS ALGORITHMS

Although the continuous-time dynamic average consensus algorithms described in the previous section are amenable to elegant and relatively simple analysis, implementing these algorithms on practical cyberphysical systems requires continuous communication between agents. This requirement is not feasible in practice due to constraints on the communication bandwidth. To address this issue, the discrete-time dynamic average consensus algorithms where the communication among agents occurs only at discrete-time steps are studied.

The main difference between continuous-time and discrete-time dynamic average consensus algorithms is the rate at which their estimates converge to the average of the reference signals. In continuous time, the parameters may be scaled to achieve any desired convergence rate, whereas in discrete time, the parameters must be carefully chosen to ensure convergence. The problem of optimizing the convergence rate has received significant attention in the literature [56]–[65]. Here, a simple method using root locus techniques for choosing the parameters to optimize the convergence rate is provided. It is also shown how to further accelerate the convergence by introducing extra dynamics into the dynamic average consensus algorithm.

The convergence rate of four discrete-time dynamic average consensus algorithms is analyzed in this section, beginning with the discretized version of the continuous-time algorithm (16). It is then shown how to use extra dynamics to accelerate the convergence rate and/or obtain robustness to initial conditions. Table 2 summarizes the arguments of the driving command of these algorithms in (2) and their special initialization requirements.

For simplicity of exposition, assume the communication graph is constant, connected, and undirected. The Laplacian matrix is then symmetric and therefore has real eigenvalues. Because the graph is connected, the smallest eigenvalue is $\lambda_1 = 0$, and all other eigenvalues are strictly positive, that is, $\lambda_2 > 0$. Furthermore, assume that the smallest and largest nonzero eigenvalues are known (if the exact eigenvalues are unknown, it also suffices to have lower and upper bounds, respectively, on λ_2 and λ_N).

Nonrobust Dynamic Average Consensus Algorithms

First consider the discretized version of the continuous-time dynamic average consensus algorithm in (16) ("Euler Discretizations of Continuous-Time Dynamic Average Consensus Algorithms" elaborates on the method for discretization and the associated range of admissible step sizes). This algorithm has the iterations

$$p_{k+1}^i = p_k^i + k_I \sum_{j=1}^{N} a_{ij}(x_k^i - x_k^j), \quad p_0^i \in \mathbb{R}, \quad i \in \{1,\ldots,N\}, \tag{27a}$$

$$x_k^i = \mathrm{u}_k^i - p_k^i, \tag{27b}$$

where $k_I \in \mathbb{R}$ is the step size. The block diagram is provided in Figure 13(a).

For discrete-time LTI systems, the convergence rate is given by the maximum magnitude of the system poles. The poles are the roots of the characteristic equation, which for the dynamic average consensus algorithm in Figure 13(a) is

$$0 = z\mathbf{I} - (\mathbf{I} - k_I\mathbf{L}).$$

If the Laplacian matrix can be diagonalized, then the system can be separated according to the eigenvalues of $\mathbf{L}$ and each subsystem analyzed separately. The characteristic equation corresponding to the eigenvalue λ of $\mathbf{L}$ is then

$$0 = 1 + \lambda\frac{k_I}{z-1}. \tag{28}$$

To observe how the pole moves as a function of the Laplacian eigenvalue, root locus techniques from LTI systems theory can be used. Figure 14(a) shows the root locus of (28) as a function of λ. The dynamic average consensus algorithm poles are then the points on the root locus at gains λ_i for $i \in \{1,\ldots,N\}$, where λ_i are the eigenvalues of the graph Laplacian. To optimize the convergence rate, the system is designed to minimize ρ, where all poles corresponding to disagreement directions (that is, those orthogonal to the consensus direction $\mathbf{1}_N$) are inside the circle centered at the origin of radius ρ. Because the pole starts at $z = 1$ and moves left as λ increases, the convergence rate is

TABLE 2 The arguments of the driving command in (2) for the reviewed discrete-time dynamic average consensus algorithms together with their initialization requirements.

Algorithm	**(27)**	**(29)**	**(30)**	**(31)**
$J^i(t)$	$\{\mathrm{u}_k^i, p_k^i\}$	$\{\mathrm{u}_k^i, p_k^i, p_{k-1}^i\}$	$\{\mathrm{u}_k^i, p_k^i, q_k^i\}$	$\{\mathrm{u}_k^i, p_k^i, p_{k-1}^i, q_k^i, q_{k-1}^i\}$
$\{I^j(t)\}_{j\in\mathcal{N}_{\mathrm{out}}^i}$	$\{x_k^j\}_{j\in\mathcal{N}_{\mathrm{out}}^i}$	$\{x_k^j\}_{j\in\mathcal{N}_{\mathrm{out}}^i,}$	$\{x_k^j, p_k^j\}_{j\in\mathcal{N}_{\mathrm{out}}^i}$	$\{x_k^j, p_k^j\}_{j\in\mathcal{N}_{\mathrm{out}}^i}$
Initialization requirement	$\sum_{j=1}^{N} p_0^j = 0$	$\sum_{j=1}^{N} p_0^j = 0$	None	None

Euler Discretizations of Continuous-Time Dynamic Average Consensus Algorithms

The continuous-time algorithms described in the article can also give rise to discrete-time strategies. Here, we describe how to discretize them so that they are implementable over wireless communication channels. This can be done by using the (forward) Euler discretization of the derivatives

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}(k+1)-\mathbf{x}(k)}{\delta},$$

where $\delta \in \mathbb{R}_{>0}$ is the step size. To illustrate the discussion, we develop this approach for (25) over a connected graph topology. The following discussion can also be extended to include iterative forms of the other continuous-time algorithms studied in the article. Using the Euler discretization in (25) leads to

$$v^i(k+1) = v^i(k) + \delta\alpha\beta \sum_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)), \tag{S14a}$$

$$\begin{aligned} x^i(k+1) &= x^i(k) + \Delta \mathrm{u}^i(k) - \delta\alpha(x^i(k) - \mathrm{u}^i(k)) \\ &\quad - \delta\beta \sum_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)) - \delta v^i(k), \end{aligned} \tag{S14b}$$

where $\Delta \mathrm{u}^i(k) = \mathrm{u}^i(k+1) - \mathrm{u}^i(k)$. To implement this iterative form at each time step k, access to the future value of the reference input at time step $k+1$ is needed. Such a requirement is not practical when the reference input is sampled from a physical process or is a result of another online algorithm. This requirement can be circumvented using a backward Euler discretization, but the resulting algorithm tracks the reference dynamic average with one-step delay. A practical solution that avoids requiring the future values of the reference input is obtained by introducing an intermediate variable $z^i(k) = x^i(k) - \mathrm{u}^i(k)$ and representing the iterative algorithm (S14) in the form

$$v^i(k+1) = v^i(k) + \delta\alpha\beta \sum_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)), \tag{S15a}$$

$$z^i(k+1) = z^i(k) - \delta\alpha z^i(k) - \delta\beta \sum_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)) - \delta v^i(k), \tag{S15b}$$

$$x^i(k) = z^i(k) + \mathrm{u}^i(k), \tag{S15c}$$

for $i \in \{1,\dots,N\}$. Equation (S15) is then implementable without the use of future inputs.

The question then is to characterize the adequate step sizes that guarantee that the convergence properties of the continuous-time algorithm are retained by its discrete implementation. Intuitively, the smaller the step size, the better for this purpose. However, this also requires more communication. To ascertain this issue, the following result is particularly useful.

Lemma S1 : Admissible Step Size for the Euler Discretized Form of Linear Time-Invariant Systems and a Bound on Their Trajectories

Consider

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad t \in \mathbb{R}_{\geq 0},$$

and its Euler discretized iterative form

$$\mathbf{x}(k+1) = (\mathbf{I} + \delta\mathbf{A})\mathbf{x}(k) + \delta\mathbf{B}\mathbf{u}(k), \quad k \in \mathbb{Z}_{\geq 0}, \tag{S16}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ are, respectively, state and input vectors, and $\delta \in \mathbb{R}_{>0}$ is the discretization step size. Let the system matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n\times n}$ be a Hurwitz matrix with eigenvalues $\{\mu_i\}_{i=1}^n$, and the difference of the input signal be bounded, $\|\Delta\mathbf{u}\| < \rho < \infty$. For any $\delta \in (0,\bar{d})$ where

$$\bar{d} = \min\left\{-2\frac{\mathrm{Re}(\mu_i)}{|\mu_i|^2}\right\}_{i=1}^{n}, \tag{S17}$$

the eigenvalues of $(\mathbf{I} + \delta\mathbf{A})$ are all located inside the until circle in the complex plane. Moreover, starting from any $\mathbf{x}(0) \in \mathbb{R}^n$, the trajectories of (S16) satisfy

$$\lim_{k\to\infty}\|\mathbf{x}(k+1)\| \leq \frac{\kappa\rho\|\mathbf{B}\|}{1-\omega}, \tag{S18}$$

where $\omega \in (0,1)$, and $\kappa \in \mathbb{R} > 0$ such that $\|\mathbf{I} + \delta\mathbf{A}\|^k \leq \kappa\omega^k$. □

The bounds $\omega \in (0,1)$ and $\kappa \in \mathbb{R}_{>0}$ in $\|\mathbf{I} + \delta\mathbf{A}\|^k \leq \kappa\omega^k$ when all the eigenvalues of $\mathbf{I} + \delta\mathbf{A}$ are located in the unit circle of the complex plane can be obtained from the following linear matrix inequality optimization problem (see [S21, Theorem 23.3] for details):

$$\begin{aligned} (\omega,\kappa,\mathbf{Q}) &= \operatorname{argmin} \omega^2, \quad \text{subject to} \\ &\frac{1}{\kappa}\mathbf{I} \leq \mathbf{Q} \leq \mathbf{I}, \quad 0 < \omega^2 < 1, \quad \kappa > 1, \\ &(\mathbf{I} + \delta\mathbf{A})^\top\mathbf{Q}(\mathbf{I} + \delta\mathbf{A}) - \mathbf{Q} \leq -(1-\omega^2)\mathbf{I}. \end{aligned} \tag{S19}$$

Building on Lemma S1, the next result characterizes the admissible discretization step size for (S15) and its ultimate tracking behavior.

Theorem S2: Convergence of (S15) Over Connected Graphs [19]

Let $\mathcal{G}$ be a connected, undirected graph. Assume that the differences of the inputs of the network satisfy $\max_{k\in\mathbb{Z}_{\geq 0}}\|(\mathbf{I} - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\Delta\mathbf{u}(k)\| = \gamma < \infty$. Then, for any $\alpha,\beta > 0$, (S15) over $\mathcal{G}$ initialized at $z^i(0) \in \mathbb{R}$ and $v^i(0) \in \mathbb{R}$ such that $\Sigma_{i=1}^N v^i(0) = 0$ has bounded trajectories that satisfy

$$\lim_{k\to\infty}|x^i(k) - \mathrm{u}^{\mathrm{avg}}(k)| \leq \frac{\delta\kappa\gamma}{1-\omega}, \quad i \in \{1,\dots,N\} \tag{S20}$$

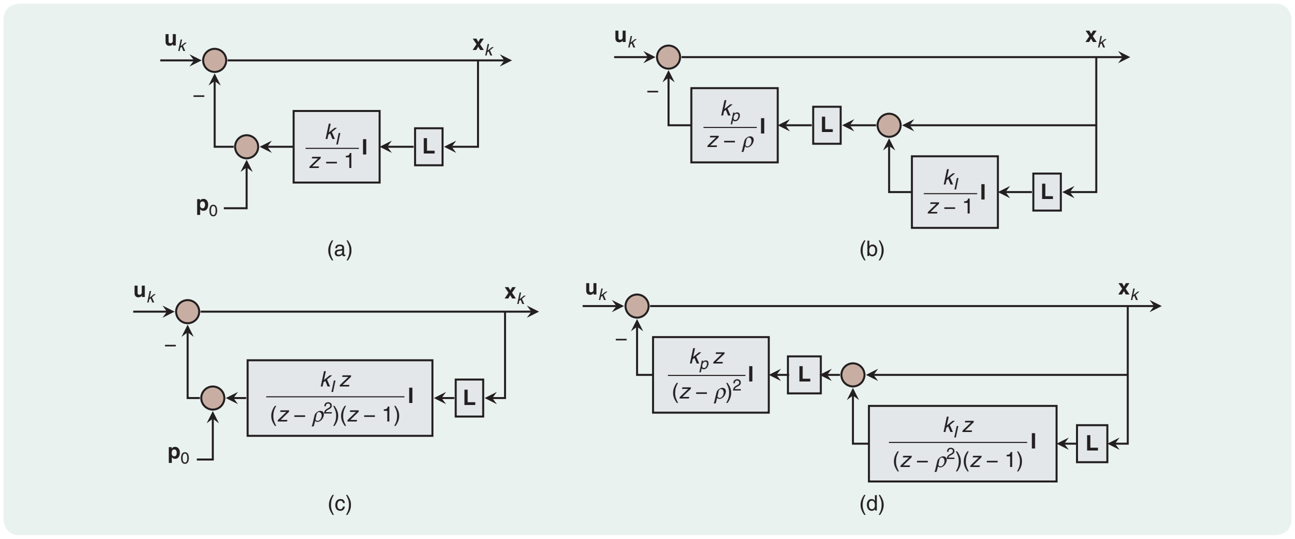
provided $\delta \in (0, \min\{\alpha^{-1}, 2\beta^{-1}(\lambda_N)^{-1}\})$. Here, λ_N is the largest eigenvalue of the Laplacian, and $\omega \in (0,1)$ and $\kappa \in \mathbb{R}_{>0}$ satisfy $\|\mathbf{I} - \delta\,\beta\mathbf{R}^\top\mathbf{L}\mathbf{R}\|^k \leq \kappa\omega^k, k \in \mathbb{Z}_{\geq 0}$. □

Note that the characterization of the step size requires knowledge of the largest eigenvalue λ_N of the Laplacian. Because such knowledge is not readily available to the network unless dedicated distributed algorithms are introduced to compute it, [19] provides the sufficient characterization $\delta \in (0, \min\{\alpha^{-1}, \beta^{-1}(\mathrm{d}_{\mathrm{out}}^{\max})^{-1}\})$ along with the ultimate tracking bound

$$\lim_{k\to\infty}|x^i(k) - \mathrm{u}^{\mathrm{avg}}(k)| \leq \frac{\delta\gamma}{\beta\lambda_2}, \quad i \in \{1,\dots,N\}.$$

REFERENCE

[S21] W. J. Rugh, *Linear Systems Theory*. Englewood Cliffs, NJ: Prentice Hall, 1993.



$\mathbf{u}_k$
$\mathbf{x}_k$
–
$\frac{k_I}{z-1}\mathbf{I}$
L
$\mathbf{p}_0$
(a)
$\mathbf{u}_k$
$\mathbf{x}_k$
–
$\frac{k_p}{z-\rho}\mathbf{I}$
L
$\frac{k_I}{z-1}\mathbf{I}$
L
(b)
$\mathbf{u}_k$
$\mathbf{x}_k$
–
$\frac{k_I z}{(z-\rho^2)(z-1)}\mathbf{I}$
L
$\mathbf{p}_0$
(c)
$\mathbf{u}_k$
$\mathbf{x}_k$
–
$\frac{k_p z}{(z-\rho)^2}\mathbf{I}$
L
$\frac{k_I z}{(z-\rho^2)(z-1)}\mathbf{I}$
L
(d)

FIGURE 13 A block diagram of discrete-time dynamic average consensus algorithms. The algorithms in (b) and (d) use proportional-integral (PI) dynamics to obtain robustness to initial conditions, whereas those in (c) and (d) use extra dynamics to accelerate the convergence rate. When the graph is connected and balanced and upper and lower bounds on the nonzero eigenvalues of the graph Laplacian are known, closed-form solutions for the parameters that optimize the convergence rate are known (see Theorem 5). (a) The nonrobust, nonaccelerated dynamic average consensus algorithm (27). (b) The robust, nonaccelerated, PI dynamic average consensus algorithm (30). (c) The nonrobust, accelerated, dynamic average consensus algorithm (29). (d) The robust, accelerated, PI dynamic average consensus algorithm (31).

optimized when there is a pole at $z=\rho$ when $\lambda=\lambda_2$ and at $z=-\rho$ when $\lambda=\lambda_N$, that is,

$$0=1+\lambda_2\frac{k_I}{\rho-1} \quad \text{and} \quad 0=1+\lambda_N\frac{k_I}{-\rho-1}.$$

Solving these conditions for k_I and ρ gives

$$k_I=\frac{2}{\lambda_2+\lambda_N} \quad \text{and} \quad \rho=\frac{\lambda_N-\lambda_2}{\lambda_N+\lambda_2}.$$

While the previous choice of parameters optimizes the convergence rate, even faster convergence can be achieved by introducing extra dynamics into the dynamic average consensus algorithm. Consider the accelerated dynamic average consensus algorithm in Figure 13(c), given by

$$p_{k+1}^i=(1+\rho^2)p_k^i-\rho^2 p_{k-1}^i+k_I\sum_{j=1}^{N}a_{ij}(x_k^i-x_k^j),$$
$$p_0^i\in\mathbb{R},\quad i\in\{1,\dots,N\}, \qquad (29a)$$
$$x_k^i=u_k^i-p_k^i. \qquad (29b)$$

Instead of a simple integrator, the transfer function in the feedback loop now has two poles (one of which is still at $z=1$). To implement the dynamic average consensus algorithm, each agent must track two internal state variables (p_k^i and p_{k-1}^i). This small increase in memory, however, can result in a significant improvement in the rate of convergence, as discussed next.

Once again, root locus techniques can be used to design the parameters to optimize the convergence rate. Figure 14(b) shows the root locus of the accelerated dynamic average consensus algorithm (29). By adding an open-loop pole at $z=\rho^2$ and zero at $z=0$, the root locus now goes around the ρ circle. Similar to the previous case, the convergence rate is optimized when there is a repeated pole at $z=\rho$ when $\lambda=\lambda_2$ and a repeated pole at $z=-\rho$ when $\lambda=\lambda_N$. This gives the optimal parameter k_I and convergence rate ρ given by

$$k_I=\frac{4}{(\sqrt{\lambda_2}+\sqrt{\lambda_N})^2} \quad \text{and} \quad \rho=\frac{\sqrt{\lambda_N}-\sqrt{\lambda_2}}{\sqrt{\lambda_N}+\sqrt{\lambda_2}}.$$

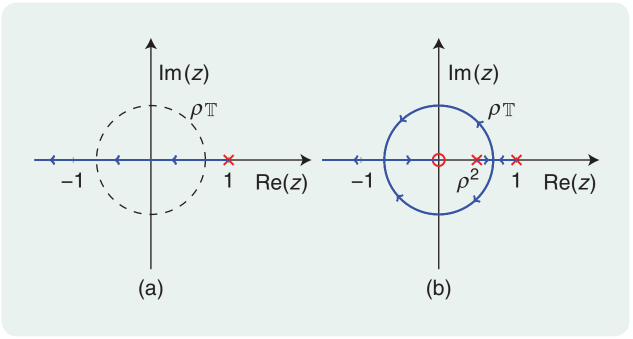


FIGURE 14 The root locus design of dynamic average consensus algorithms. The dynamic average consensus algorithm poles are the points on the root locus at gains λ_i for $i\in\{1,\dots,N\}$, where λ_i are the eigenvalues of the graph Laplacian. To optimize the convergence rate, the parameters are chosen to minimize ρ such that all poles corresponding to eigenvalues λ_i for $i\in\{2,\dots,N\}$ are inside the circle centered at the origin of radius ρ. The dynamic average consensus algorithm then converges linearly with rate ρ. (a) The accelerated, dynamic average consensus algorithm in Figure 13(a). (b) The nonaccelerated, dynamic average consensus algorithm in Figure 13(c).

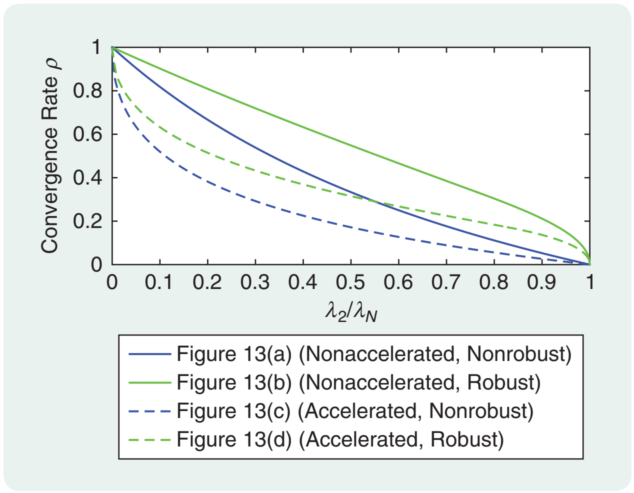


FIGURE 15 The convergence rate ρ as a function of λ_2/λ_N for the dynamic average consensus algorithms in Figure 13. The accelerated dynamic average consensus algorithms (dashed lines) use extra dynamics to enhance the convergence rate, as opposed to the nonaccelerated algorithms (solid lines). Also, the robust algorithms (green) use the proportional-integral structure to obtain robustness to initial conditions as opposed to the nonrobust algorithms (blue). The graph is assumed to be constant, connected, and undirected with Laplacian eigenvalues λ_i for $i \in \{1,\dots,N\}$. Closed-form expressions for the rates and algorithm parameters are provided in Theorem 5.

The convergence rates of both the standard (27) and accelerated (29) versions of the dynamic average consensus algorithm are plotted in Figure 15 as a function of the ratio λ_2/λ_N.

Robust Dynamic Average Consensus Algorithms

Although the previous dynamic average consensus algorithms are not robust to initial conditions, root locus techniques can also be used to optimize the convergence rate of dynamic average consensus algorithms that are robust to initial conditions. Consider the discrete-time version of the PI estimator from (19), whose iterations are given by

$$q_{k+1}^i = \rho q_k^i + k_p \sum_{j=1}^{N} a_{ij}((x_k^i - x_k^j) + (p_k^i - p_k^j)), \quad (30a)$$

$$p_{k+1}^i = p_k^i + k_I \sum_{j=1}^{N} a_{ij}(x_k^i - x_k^j), \quad (30b)$$

$$x_k^i = u_k^i - q_k^i, \qquad p_0^i, q_0^i \in \mathbb{R}, \qquad i \in \{1,\dots,N\}, \quad (30c)$$

with parameters $\rho, k_p, k_I \in \mathbb{R}$. The block diagram of this algorithm is shown in Figure 13(b).

Because the dynamic average consensus algorithms (27) and (29) have only one Laplacian block in the block diagram, the resulting root loci are linear in the Laplacian eigenvalues. For the PI dynamic average consensus algorithm, however, the block diagram contains two Laplacian blocks, resulting in a quadratic dependence on the eigenvalues. Instead of a linear root locus, the design involves a quadratic root locus. Although this complicates the design process, closed-form solutions for the algorithm parameters can still be found [57], even for the accelerated version using extra dynamics, given by

$$q_{k+1}^i = 2\rho q_k^i - \rho^2 q_{k-1}^i + k_p \sum_{j=1}^{N} a_{ij}((x_k^i - x_k^j) + (p_k^i - p_k^j)), \quad (31a)$$

$$p_{k+1}^i = (1+\rho^2) p_k^i - \rho^2 p_{k-1}^i + k_I \sum_{j=1}^{N} a_{ij}(x_k^i - x_k^j), \quad (31b)$$

$$x_k^i = u_k^i - q_k^i, \qquad p_0^i, q_0^i \in \mathbb{R}, \qquad i \in \{1,\dots,N\}, \quad (31c)$$

whose block diagram is in Figure 13(d). The resulting convergence rate is plotted in Figure 15. Although the convergence rates of the standard and accelerated PI dynamic average consensus algorithms are slower than those of (27) and (29), respectively, they have the additional advantage of being robust to initial conditions.

The following result summarizes the parameter choices that optimize the convergence rate for each of the discrete-time dynamic average consensus algorithms in Figure 13. The results for the first two algorithms follow from the previous discussion, whereas details of the results for the last two algorithms can be found in [57].

Theorem 5: Optimal Convergence Rates of Discrete-Time Dynamic Average Consensus Algorithms

Let $\mathcal{G}$ be a connected, undirected graph. Suppose the reference signal $\mathbf{u}^i$ at each agent $i \in \{1,\dots,N\}$ is a constant scalar. Consider the dynamic average consensus algorithms in Figure 13, with the parameters chosen according to Table 3 [the algorithms in Figure 13(a) and (c) are initialized such that the average of the initial integrator states is zero, that is, $\Sigma_{i=1}^N p_0^i = 0$]. The agreement states $x_k^i, i \in \{1,\dots,N\}$ converge to $\mathbf{u}^{\text{avg}}$ exponentially with rate ρ. ☐

PERFECT TRACKING USING A PRIORI KNOWLEDGE OF THE INPUT SIGNALS

The design of the dynamic average consensus algorithms described in the discussion so far does not require prior knowledge of the reference signals and is therefore broadly applicable. This also comes at a cost. The convergence guarantees of these algorithms are strong only when the reference signals are constant or slowly varying. The error of such algorithms can be large, however, when the reference signals change quickly in time. This section describes dynamic average consensus algorithms, which are capable of tracking fast time-varying signals with either zero or small steady-state error. In each case, their design assumes some specific information about the nature of the reference signals. In particular, consider reference signals that 1) have a known model, 2) are band limited, or 3) have bounded derivatives.

TABLE 3 The parameter selection for the dynamic average consensus algorithms of Figure 13 as a function of the minimum and maximum nonzero Laplacian eigenvalues λ_2 and λ_N, respectively, with $\lambda_r := \lambda_2/\lambda_N$. N/A: not applicable.

	ρ	k_I	k_p
Figure 13(a)	$\frac{\lambda_N-\lambda_2}{\lambda_N+\lambda_2}$	$\frac{2}{\lambda_2+\lambda_N}$	N/A
Figure 13(b)	$\begin{cases} \frac{8-8\lambda_r+\lambda_r^2}{8-\lambda_r^2}, & 0<\lambda_r\le 3-\sqrt{5} \\ \frac{\sqrt{(1-\lambda_r)(4+\lambda_r^2(5-\lambda_r))}-\lambda_r(1-\lambda_r)}{2(1+\lambda_r^2)}, & 3-\sqrt{5}<\lambda_r\le 1 \end{cases}$	$\frac{1-\rho}{\lambda_2}$	$\frac{1}{\lambda_N}\frac{\rho(1-\rho)\lambda_r}{\rho+\lambda_r-1}$
Figure 13(c)	$\frac{\sqrt{\lambda_N}-\sqrt{\lambda_2}}{\sqrt{\lambda_N}+\sqrt{\lambda_2}}$	$\frac{4}{(\sqrt{\lambda_2}+\sqrt{\lambda_N})^2}$	N/A
Figure 13(d)	$\begin{cases} \frac{6-2\sqrt{1-\lambda_r}+\lambda_r-4\sqrt{2-2\sqrt{1-\lambda_r}+\lambda_r}}{2+2\sqrt{1-\lambda_r}-\lambda_r}, & 0<\lambda_r\le 2(\sqrt{2}-1) \\ \frac{-3-2\sqrt{1-\lambda_r}+\lambda_r+2\sqrt{2+2\sqrt{1-\lambda_r}-\lambda_r}}{-1-2\sqrt{1-\lambda_r}+\lambda_r}, & 2(\sqrt{2}-1)<\lambda_r\le 1 \end{cases}$	$\frac{(1-\rho)^2}{\lambda_2}$	$(2+2\sqrt{1-\lambda_r}-\lambda_r)k_I$

Signals With a Known Model (Discrete Time)

The discrete-time dynamic average consensus algorithms discussed previously are designed with the idea of tracking constant reference signals with zero steady-state error. To do this, the algorithms contain an integrator in the feedback loop. This concept generalizes to time-varying signals with a known model using the *internal model principle*. Consider reference signals whose z-transform has the form $\mathsf{u}^i(z) = n^i(z)/d(z)$, where $n^i(z)$ and $d(z)$ are polynomials in z for $i \in \{1,\dots,N\}$. Dynamic average consensus algorithms can be designed to have zero steady-state error for such signals by placing the model of the input signals [that is, $d(z)$] in the feedback loop. Some common examples of models are

$$d(z) = \begin{cases} (z-1)^m, & \text{polynomial of degree } m-1 \\ z^2 - 2z\cos(\omega)+1, & \text{sinusoid with frequency } \omega. \end{cases}$$

This section focuses on dynamic average consensus algorithms that track degree $m-1$ polynomial reference signals with zero steady-state error.

Consider the dynamic average consensus algorithms in Figure 16. The transfer function of each algorithm has m zeros at $z=1$, so the algorithms track degree $m-1$ polynomial references signals with zero steady-state error. The time-domain equations for the dynamic average consensus algorithm in Figure 16(a) are

$$x^i_{1,k+1} = x^i_{1,k} - \sum_{j=1}^N a_{ij}(x^i_{1,k} - x^j_{1,k}) + \Delta^{(m)}\mathsf{u}_k, \tag{32a}$$

$$x^i_{2,k+1} = x^i_{2,k} - \sum_{j=1}^N a_{ij}(x^i_{2,k} - x^j_{2,k}) + x^i_{1,k}, \tag{32b}$$

$$\vdots$$

$$x^i_{m,k+1} = x^i_{m,k} - \sum_{j=1}^N a_{ij}(x^i_{m,k} - x^j_{m,k}) + x^i_{m-1,k}, \tag{32c}$$

$$x^i_k = x^i_{m,k}, \quad x^i_{\ell,0}\in\mathbb{R}, \quad \ell\in\{1,\dots,m\}, \quad i\in\{1,\dots,N\}, \tag{32d}$$

where the mth divided difference is defined recursively as $\Delta^{(m)}\mathsf{u}^i_k = \Delta^{(m-1)}\mathsf{u}^i_k - \Delta^{(m-1)}\mathsf{u}^i_{k-1}$ for $m \ge 2$ with $\Delta^{(1)}\mathsf{u}^i_k = \mathsf{u}^i_k - \mathsf{u}^i_{k-1}$. The estimate of the average, however, is delayed by m iterations due to the transfer function having a factor of z^{-m} between the input and output. This problem is fixed by the dynamic average consensus algorithm in Figure 16(b), given by

$$p^i_{1,k+1} = p^i_{1,k} + \sum_{j=1}^N a_{ij}((\mathsf{u}^i_k - \mathsf{u}^j_k) - (p^i_{1,k} - p^j_{1,k})), \tag{33a}$$

$$p^i_{2,k+1} = p^i_{2,k} + \sum_{j=1}^N a_{ij}((\mathsf{u}^i_k - \mathsf{u}^j_k) - (p^i_{1,k} - p^j_{1,k}) - (p^i_{2,k} - p^j_{2,k})), \tag{33b}$$

$$\vdots$$

$$p^i_{m,k+1} = p^i_{m,k} + \sum_{j=1}^N a_{ij}((\mathsf{u}^i_k - \mathsf{u}^j_k) - \sum_{\ell=1}^m (p^i_{\ell,k} - p^j_{\ell,k})), \tag{33c}$$

$$x^i_k = u^i_k - \sum_{\ell=1}^m p^i_{\ell,k}, \quad p^i_{\ell,0}\in\mathbb{R}, \quad \ell\in\{1,\dots,m\}, \quad i\in\{1,\dots,N\}, \tag{33d}$$

which tracks degree $m-1$ polynomial reference signals with zero steady-state error without delay. Note, however, that the communication graph is assumed to be constant to use frequency-domain arguments; although the output of the dynamic average consensus algorithm in Figure 16(a) is delayed, it also has nice tracking properties when the communication graph is time varying, whereas the dynamic average consensus algorithm in Figure 16(b) does not.

To track degree $m-1$ polynomial reference signals, each dynamic average consensus algorithm in Figure 16 cascades m dynamic average consensus algorithms, each with a pole at $z=1$ in the feedback loop. The dynamic average consensus algorithm (27) is cascaded in Figure 16(b), but any of the dynamic average consensus algorithms from the previous section could also be used. For example, the PI dynamic average consensus algorithm could be cascaded m times to track degree $m-1$ polynomial reference signals with zero steady-state error independent of the initial conditions.

In general, reference signals with model $d(z)$ can be tracked with zero steady-state error by cascading simple dynamic average consensus algorithms, each of which tracks a factor of $d(z)$. In particular, suppose $d(z) = d_1(z)d_2(z)\ldots d_m(z)$. Then, m dynamic average consensus algorithms can be cascaded, where the ith component contains the model $d_i(z)$ for $i = 1,\ldots,m$. Alternatively, a single dynamic average consensus algorithm can be designed that contains the entire model $d(z)$. This approach using an internal model version of the PI dynamic average consensus algorithm is designed in [66] in both continuous time and discrete time.

In many practical applications, the exact model of the reference signals is unknown. However, it is shown in [67] that a frequency estimator can be used in conjunction with an internal model dynamic average consensus algorithm to still achieve zero steady-state error. In particular, the frequency of the reference signals is estimated such that the estimate converges to the actual frequency. This time-varying estimate of the frequency is then used in place of the true frequency to design the feedback dynamic average consensus algorithm [67].

Band-Limited Signals (Discrete Time)

To use algorithms designed using the model of the reference signals, the signals must be composed of a finite number of known frequencies. When either the frequencies are unknown or there are infinitely many frequencies, dynamic average consensus algorithms can still be designed if the reference signals are band limited. In this case, feedforward dynamic average consensus algorithm designs can be used to achieve arbitrarily small steady-state error.

For this discussion, assume that the reference signals are band limited with known cutoff frequency. In particular, let $\mathsf{U}^i(z)$ be the z-transform of the ith reference signal $\{\mathsf{u}_k^i\}$. Then $\mathsf{U}^i(z)$ is band limited with cutoff frequency θ_c if $|\mathsf{U}^i(\exp(j\theta))| = 0$ for all $\theta \in (\theta_c, \pi]$.

Consider the dynamic average consensus algorithm in Figure 17. The reference signals are passed through a prefilter $h(z)$ and then multiplied m times by the consensus matrix $\mathbf{I} - \mathbf{L}$ with a delay between each (to allow time for communication). The transfer function from the input $\mathbf{U}(z)$ to the output $\mathbf{X}(z)$ is

$$H(z,\mathbf{L}) = h(z)\frac{1}{z^m}(\mathbf{I}-\mathbf{L})^m.$$

For the tracking error to be small, $h(z)$ must approximate z^m for all $\theta \in [0,\theta_c]$, where $z = \exp(j\theta)$ and θ_c is the cutoff frequency. In this case, the transfer function in the passband is approximately

$$H(z,\mathbf{L}) \approx (\mathbf{I}-\mathbf{L})^m,$$

so the error can be made small by making m large enough (so long as $\mathbf{L}$ is scaled such that $\|\mathbf{I}-\mathbf{L}-\mathbf{1}\mathbf{1}^\mathsf{T}/N\|_2 < 1$).

Specifically, the prefilter is designed such that $h(z)$ is proper and $h(z) \approx z^m$ for $z = \exp(j\theta)$ for all $\theta \in [0,\theta_c]$ [note that $h(z) = z^m$ cannot be used because it is not causal]. An m-step filter can be obtained by cascading a one-step filter m

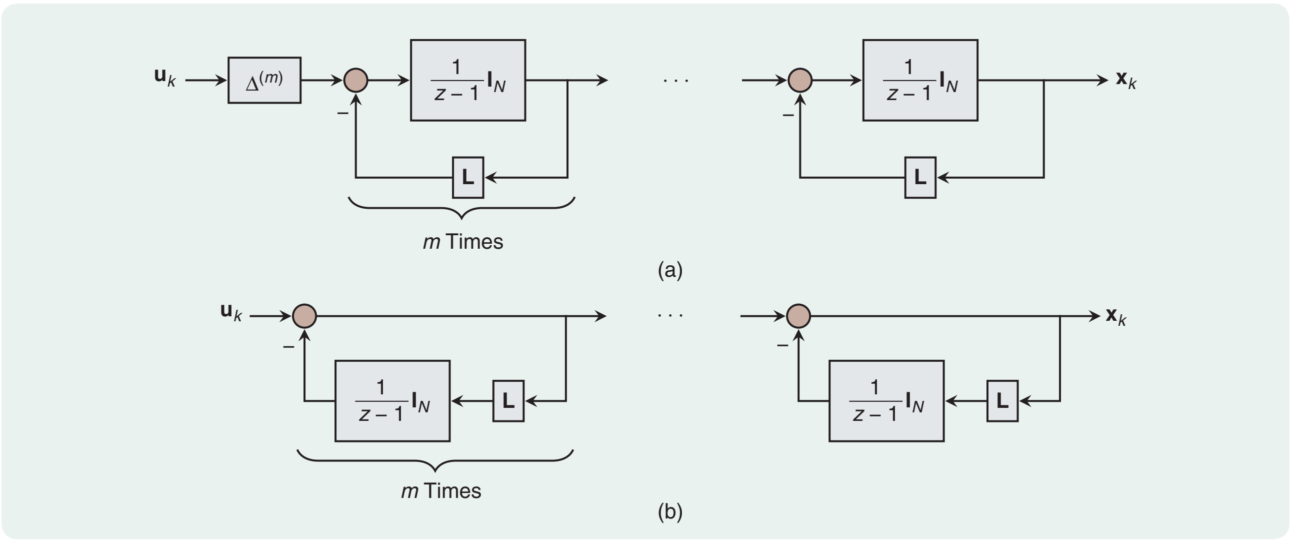


FIGURE 16 A block diagram of dynamic average consensus algorithms that track polynomial signals of degree $m-1$ with zero steady-state error when initialized correctly (neither algorithm is robust to initial conditions). The indicated section is repeated in series m times. (a) The performance of (32) does not degrade when the graph is time varying, but the estimate is delayed by m iterations. Furthermore, the algorithm is numerically unstable when m is large and eventually diverges from tracking the average when implemented using finite precision arithmetic. (b) The estimate of the average by (33) is not delayed, and the algorithm is numerically stable, but the tracking performance degrades when the communication graph is time varying. (a) Dynamic average consensus algorithm (32) in [76] where $\Delta^{(m)} = (1-z^{-1})^m$ is the mth divided difference (see also [77] for a step-size analysis). (b) Dynamic average consensus algorithm (33), which is the algorithm in [53] cascaded in series m times.

times in series. In other words, let $h(z) = [zf(z)]^m$, where $f(z)$ is strictly proper and approximates unity in the passband. Because $f(z)$ must approximate unity in both magnitude *and* phase, a standard lowpass filter cannot be used. Instead, set

$$f(z) = 1 - \frac{g(z)}{\lim_{z\to\infty} g(z)},$$

where $g(z)$ is a proper high-pass filter with cutoff frequency θ_c. Then $f(z)$ is strictly proper (due to the normalizing constant in the denominator) and approximates unity in the band $[0, \theta_c]$ [because $g(z)$ is high pass]. Therefore, a prefilter $h(z)$ that approximates z^m in the passband can be designed using a standard high-pass filter $g(z)$.

Using such a prefilter, [68] makes the error of the dynamic average consensus algorithm in Figure 17 arbitrarily small if 1) the graph is connected and balanced at each time step (in particular, it need not be constant), 2) $\mathbf{L}$ is scaled such that $\| \mathbf{I} - \mathbf{L} - \mathbf{1}\mathbf{1}^\mathsf{T}/N \|_2 < 1$, 3) the number of stages m is made large enough, 4) the prefilter can approximate z^m arbitrarily closely in the passband, and 5) exact arithmetic is used. Note that exact arithmetic is required for arbitrarily small errors because rounding errors cause high-frequency components in the reference signals.

Signals With Bounded Derivatives (Continuous Time)

Stronger tracking results can be obtained using algorithms implemented in continuous time. Here, a number of continuous-time dynamic average consensus algorithms are presented that are capable of tracking time-varying reference signals whose derivatives are bounded with zero error in *finite time*. For simplicity, assume that the communication graph is constant, connected, and undirected. Also, the reference signals are assumed to be differentiable with bounded derivatives.

In discrete time, zero steady-state error is obtained by placing the internal model of the reference signals in the feedback loop. This provides infinite loop gain at the frequencies contained in the reference signals. In continuous time, however, the discontinuous signum function sgn can be used in the feedback loop to provide infinite loop gain over all frequencies, so no model of the reference signals is required. Furthermore, such continuous-time dynamic average consensus algorithms are capable of achieving zero error tracking in finite time as opposed to the exponential convergence achieved by discrete-time dynamic average consensus algorithms. One such algorithm is described in [69] as

$$\dot{x}^i = \dot{\mathsf{u}}^i - k_p \sum_{j\in\mathcal{N}^i_{\text{out}}} \operatorname{sgn}(x^i(t) - x^j(t)), \qquad i \in \{1,\dots,N\}, \tag{34a}$$

$$\sum_{j=1}^{N} x^j(0) = \sum_{j=1}^{N} \mathsf{u}^j(0). \tag{34b}$$

The block diagram representation in Figure 18(a) indicates that this algorithm applies sgn in the feedback loop. Under the given assumptions, using a sliding mode argument, the feedback gain k_p can be selected to guarantee zero error tracking in finite time, provided that an upper bound γ of the form $\sup_{\tau\in[0,\infty)} \| \dot{\mathbf{u}}(\tau) \| = \gamma < \infty$ is known [69]. The dynamic consensus algorithm (34) can also be implemented without derivative information of the reference signals in an equivalent way as

$$\dot{p}^i = k_p \sum_{j\in\mathcal{N}^i_{\text{out}}} \operatorname{sgn}(x^i - x^j), \qquad \sum_{j=1}^{N} p^j(0) = 0, \tag{35a}$$

$$x^i = \mathsf{u}^i - p^i, \qquad i \in \{1,\dots,N\}. \tag{35b}$$

The corresponding block diagram is shown in Figure 18(b).

It is simple to see from the block diagram of Figure 18(a) why (34) is not robust to initial conditions; the integrator state is directly connected to the output and therefore affects the steady-state output in the consensus direction. This issue is addressed by the dynamic average consensus algorithm in Figure 18(c), given by

$$\dot{p}^i = k_p \operatorname{sgn}\Big(\sum_{j\in\mathcal{N}^i_{\text{out}}} (x^i - x^j) \Big), \qquad p^i(0) \in \mathbb{R}, \tag{36a}$$

$$x^i = \mathsf{u}^i - \sum_{j\in\mathcal{N}^i_{\text{out}}} (p^i - p^j), \qquad i \in \{1,\dots,N\}, \tag{36b}$$

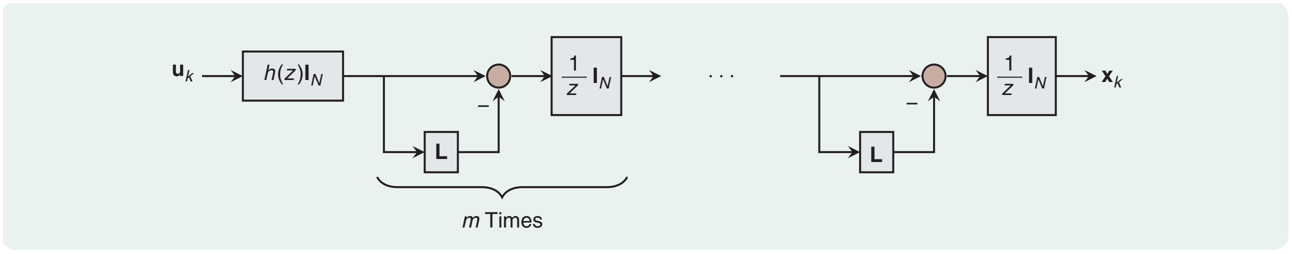


FIGURE 17 The feedforward dynamic average consensus algorithm for tracking the average of band-limited reference signals. The prefilter $h(z)$ is applied to the reference signals *before* passing through the graph Laplacian. For an appropriately designed prefilter, the dynamic average consensus algorithm can track band-limited reference signals with arbitrarily small steady-state error when using exact arithmetic (and small error for finite precision) [68].

which moves the integrator before the Laplacian in the feedback loop. However, this dynamic average consensus algorithm has two Laplacian blocks directly connected, which means that it requires two-hop communication to implement. In other words, two sequential rounds of communication are required at each time instant. In the time domain, each agent must perform the following (in order) at each time t: 1) communicate $p^i(t)$, 2) calculate $x^i(t)$, 3) communicate $x^i(t)$, and 4) update $p^i(t)$ using the derivative $\dot{p}^i(t)$. To require only one-hop communication, the dynamic average consensus algorithm in Figure 18(d), given by

$$\dot{q}^i = -\alpha q^i + x^i, \tag{37a}$$

$$\dot{p}^i = k_p \operatorname{sgn}\Big(\sum_{j\in\mathcal{N}^i_{\text{out}}} (q^i - q^j)\Big), \tag{37b}$$

$$x^i = \mathsf{u}^i - \sum_{j\in\mathcal{N}^i_{\text{out}}} (p^i - p^j), \quad p^i(0), q^i(0) \in \mathbb{R}, \quad i \in \{1,\ldots,N\}, \tag{37c}$$

places a strictly proper transfer function in the path between the Laplacian blocks. The extra dynamics, however, cause the output to converge exponentially instead of in finite time [70].

Alternatively, under the given assumptions, a sliding-mode-based dynamic average consensus algorithm with

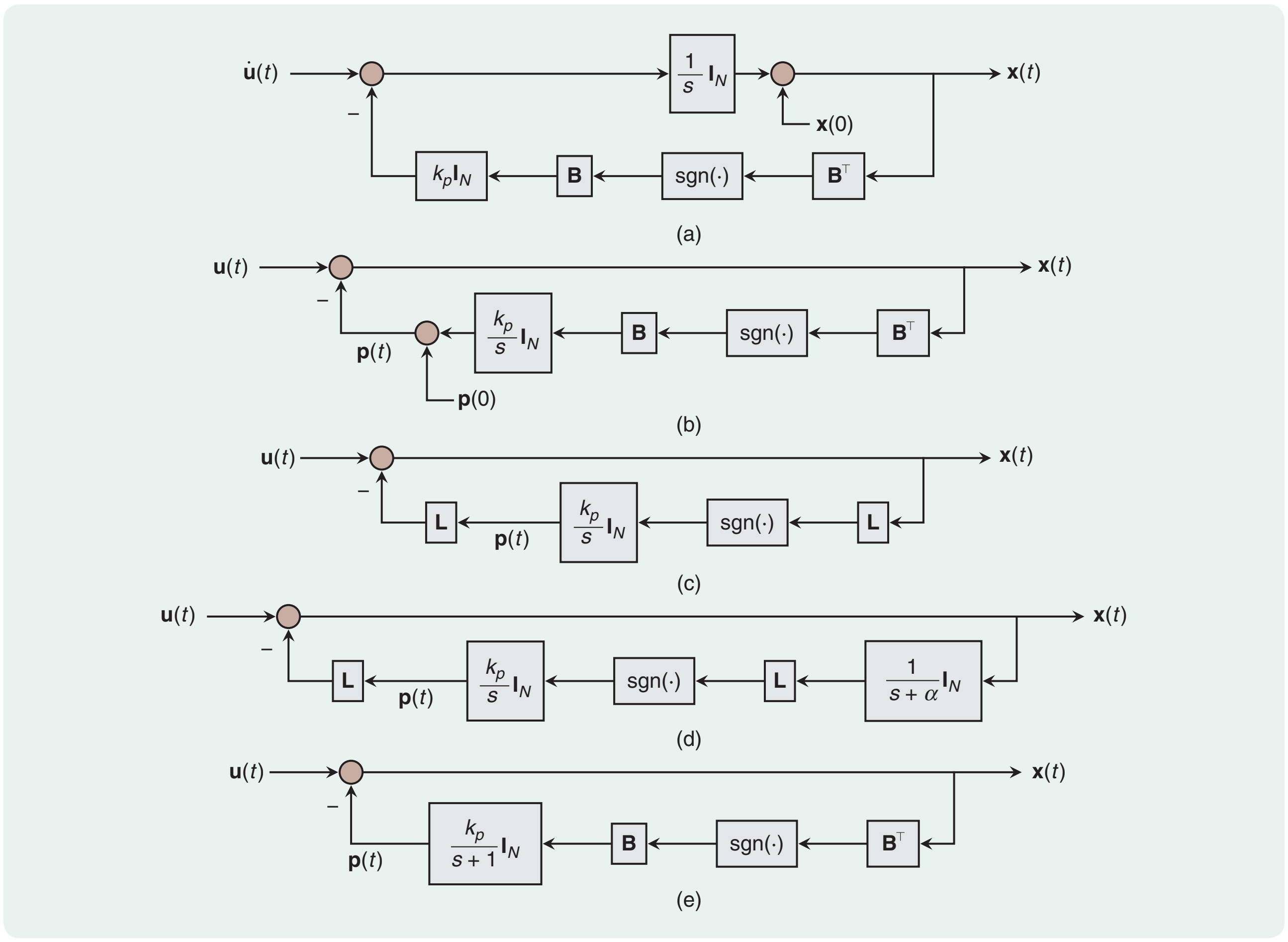


FIGURE 18 A block diagram of discontinuous dynamic average consensus algorithms in continuous time. In each case, the communication graph is assumed to be constant, connected, and balanced with Laplacian matrix $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$. Furthermore, the reference signals are assumed to have bounded derivatives. The dynamic average consensus algorithm (34), shown in (a), achieves perfect tracking in finite time and uses one-hop communication, but it is not robust to initial conditions [that is, the steady-state error is zero only if $1_N^\top \mathbf{x}(0) = 1_N^\top \mathbf{u}(0)$]. Furthermore, the derivative of the reference signals is required (see [69]). The dynamic average consensus algorithm (35) shown in (b) is equivalent to the algorithm in (a), although this form does not require the derivative of the reference signals. In this case, the requirement on the initial conditions is $1_N^\top \mathbf{p}(0) = 0$. The dynamic average consensus algorithm (36) shown in (c) converges to zero error in finite time and is robust to initial conditions but requires two-hop communication (in other words, two rounds of communication are performed at each time instant) (see [70]). The dynamic average consensus algorithm (37) shown in (d) is robust to initial conditions and uses one-hop communication but converges to zero error exponentially instead of in finite time (see [70]). The dynamic average consensus algorithm (38) shown in (e) is robust to initial conditions and uses one-hop communication, although the error converges to zero exponentially instead of in finite time (see [71]). (a) The dynamic average consensus algorithm (34). (b) The dynamic average consensus algorithm (35). (c) The dynamic average consensus algorithm (36). (d) The dynamic average consensus algorithm (37). (e) The dynamic average consensus algorithm (38).

zero error tracking, which can be arbitrarily initialized, is provided in [71] as

$$\dot{x}^i = \dot{\mathsf{u}}^i - (x^i - \mathsf{u}^i) - k_p \sum_{j\in\mathcal{N}^i_{\text{out}}} \operatorname{sgn}(x^i - x^j), \quad x^i(0) \in \mathbb{R},$$
$$i \in \{1,\ldots,N\},$$

or equivalently,

$$\dot{p}^i = -p^i + k_p \sum_{j\in\mathcal{N}^i_{\text{out}}} \operatorname{sgn}(x^i - x^j), \qquad p^i(0) \in \mathbb{R}, \tag{38a}$$
$$x^i = \mathsf{u}^i - p^i, \quad i \in \{1,\ldots,N\}. \tag{38b}$$

However, this algorithm requires both the reference signals and their derivatives to be bounded with known values γ_1 and γ_2: $\sup_{\tau\in[0,\infty)}\|\mathbf{u}(\tau)\| = \gamma_1 < \infty$ and $\sup_{\tau\in[0,\infty)}\|\dot{\mathbf{u}}(\tau)\| = \gamma_2 < \infty$. These values are required to design the proper sliding mode gain k_p.

The continuous-time finite-time algorithms described in this section exhibit a sliding mode behavior. In fact, they converge in finite time to the agreement manifold and then slide on it by switching continuously at an infinite frequency between two system structures. This phenomenon is called *chattering*. Recall that commutations at infinite frequency between two subsystems is called the *Zeno phenomenon* in the literature on hybrid systems. See [72] for a detailed discussion on the relation between first-order sliding mode chattering and the Zeno phenomenon. From a practical perspective, chattering is undesirable and leads to excessive control energy expenditure [73]. A common approach to eliminate chattering is to smooth out the control discontinuity in a thin boundary layer around the switching surface. However, this approach leads to a tracking error that is proportional to the thickness of the boundary layer. Another approach to address is the use of higher-order sliding mode control; see [74] for details. To the best of our knowledge, higher-order sliding mode control has not been used in the context of dynamic average consensus, although there exist results for other networked agreement problems [75].

CONCLUSIONS AND FUTURE DIRECTIONS

This article has provided an overview of the state of the art on the available distributed algorithmic solutions to tackle the dynamic average consensus problem. It begins by exploring several applications of dynamic average consensus in cyberphysical systems, including distributed formation control, state estimation, convex optimization, and optimal resource allocation. Using dynamic average consensus as a backbone, these advanced distributed algorithms enable groups of agents to coordinate to solve complex problems. Starting from the static consensus problem, we then derived dynamic average consensus algorithms for various scenarios. Continuous-time algorithms are first introduced (along with simple techniques for analyzing them) using block diagrams to provide intuition for the algorithmic structure. To reduce the communication bandwidth, how to choose the step sizes to optimize the convergence rate when implemented in discrete time is shown, along with how to accelerate the convergence rate by introducing extra dynamics. Finally, how to use a priori information about the reference signals to design algorithms with improved tracking performance is shown.

The goal is that the article helps the reader obtain an overview of the progress and intricacies of this topic and appreciate the design tradeoffs faced when balancing desirable properties for large-scale interconnected systems, such as convergence rate, steady-state error, robustness to initial conditions, internal stability, amount of memory required on each agent, and amount of communication between neighboring agents. Given the importance of the ability to track the average of time-varying reference signals in network systems, it is expected that the number and breadth of applications for dynamic average consensus algorithms will continue to increase in such areas as the smart grid, autonomous vehicles, and distributed robotics.

Many interesting questions and avenues for further research remain open. For instance, the emergence of opportunistic state-triggered ideas in the control and coordination of networked cyberphysical systems presents exciting opportunities for the development of novel solutions to the dynamic average consensus problem. The underlying theme of this effort is to abandon the paradigm of periodic or continuous sampling/control in exchange for deliberate, opportunistic aperiodic sampling/control to improve efficiency. Beyond the brief incursion on this topic in "Dynamic Average Consensus Algorithms with Continuous-Time Evolution and Discrete-Time Communication," further research is needed on synthesizing triggering criteria for individual agents that prescribe when information is to be shared with or acquired from neighbors, which lead to convergence guarantees and are amenable to the characterization of performance improvements over periodic discrete-time implementations.

The use of event triggering also opens up the way to employ other interesting forms of communication and computation among the agents when solving the dynamic average consensus problem, such as, for instance, the cloud. In cloud-based coordination, instead of direct peer-to-peer communication, agents interact indirectly by opportunistically communicating with the cloud to leave messages for other agents. These messages can contain information about their current estimates, future plans, or fallback strategies. The use of the cloud also opens the possibility of network agents with limited capabilities taking advantage of high-performance computation capabilities to deal with complex processes. The time-varying nature of the signals available at the individual agents in the dynamic average consensus problem raises many interesting challenges that must be addressed to take advantage of this approach. Related to the focus of this effort on the communication aspects, the development of initialization-free dynamic average consensus algorithms over directed graphs is also another important line of research.

Dynamic Average Consensus Algorithms With Continuous-Time Evolution and Discrete-Time Communication

We discuss here an alternative to the discretization route explained in "Euler Discretizations of Continuous-Time Dynamic Average Consensus Algorithms" to produce implementable strategies from the continuous-time algorithms described in the article. This approach is based on the observation that, when implementing the algorithms over digital platforms, computation can still be reasonably approximated by continuous-time evolution (given the ever-growing capabilities of modern embedded processors and computers), whereas communication is a process that still requires proper acknowledgment of its discrete-time nature. The basic idea is to opportunistically trigger, based on the network state, the times for information sharing among agents to take place and allow individual agents to determine these autonomously. This has the potential to result in more efficient algorithm implementations because performing communication usually requires more energy than computation [S22]. In addition, the use of fixed communication step sizes can lead to a wasteful use of the network resources because of the need to select it, taking into account worst-case scenarios. These observations are aligned with the ongoing research activity [S23], [S24] on event-triggered control and aperiodic sampling for controlled dynamical systems that seeks to trade computation and decision making for less communication, sensing, or actuator effort while still guaranteeing a desired level of performance. The surveys [S25], [S26] describe how these ideas can be employed to design event-triggered communication laws for static average consensus.

Motivated by these observations, [S15] investigates a discrete-time communication implementation of the continuous-time algorithm (25) for dynamic average consensus. Under this strategy, the algorithm becomes

$$\dot{v}^i = \alpha\beta\sum_{j=1}^{N} a_{ij}(\hat{x}^i - \hat{x}^j), \qquad \text{(S21a)}$$

$$\dot{x}^i = \dot{u}^i - \alpha(x^i - u^i) - \beta\sum_{j=1}^{N} a_{ij}(\hat{x}^i - \hat{x}^j) - v^i, \qquad \text{(S21b)}$$

for each $i \in \{1,\dots,N\}$, where $\hat{x}^i(t) = x^i(t_k^i)$ for $t \in [t_k^i, t_{k+1}^i)$, with $\{t_k^i\} \subset \mathbb{R}_{\geq 0}$ denoting the sequence of times at which agent i communicates with its in-neighbors. The basic idea is that agents share their information with neighbors when the uncertainty in the outdated information is such that the monotonic convergent behavior of the overall network can no longer be guaranteed. The design of such triggers is challenging because of the following requirements: 1) triggers need to be distributed so that agents can check them with the information available to them from their out-neighbors, 2) they must guarantee the absence of Zeno behavior (the undesirable situation where an infinite number of communication rounds are triggered in a finite amount of time), and 3) they have to ensure the network achieves dynamic average consensus, although agents operate with outdated information while inputs are changing with time.

Consider the following event-triggered communication law [S15]: each agent is to communicate with its in-neighbors at times $\{t_k^i\}_{k\in\mathbb{N}} \subset \mathbb{R}_{\geq 0}$, starting at $t_1^i = 0$, determined by

$$t_{k+1}^i = \operatorname{argmax}\left\{t \in [t_k^i, \infty) \,\middle|\, |x^i(t_k^i) - x^i(t)| \leq \epsilon_i\right\}. \qquad \text{(S22)}$$

Here, $\epsilon_i \in \mathbb{R}_{>0}$ is a constant value that each agent chooses locally to control its inter-event times and avoid Zeno behavior. Specifically, the interexecution times of each agent $i \in \{1,\dots,N\}$ employing (S22) are lower bounded by

$$\tau^i = \frac{1}{\alpha}\ln\left(1 + \frac{\alpha\epsilon_i}{c^i}\right), \qquad \text{(S23)}$$

where c^i and η are positive real numbers that depend on the initial conditions and network parameters (we omit for simplicity their specific form, but see [S15] for the explicit expressions). The lower bound (S23) shows that, for a positive nonzero ϵ^i, the interexecution times are bounded away from zero, and it is guaranteed that, for networks with a finite number of agents, the implementation of (S21) with the communication trigger law (S22) is Zeno free. The following result formally describes the convergence behavior of (S21) under (S22) when the interaction topology is modeled by a strongly connected and weight-balanced digraph.

Theorem S3: Convergence of (S21) Over Strongly Connected and Weight-Balanced Digraph with Asynchronous Distributed Event-Triggered Communication [S15]

Let $\mathcal{G}$ be a strongly connected and weight-balanced digraph. Assume the reference signals satisfy $\sup_{t\in[0,\infty)}|\dot{u}^i(t)| = \kappa^i < \infty$, for $i \in \{1,\dots,N\}$, and $\sup_{t\in[0,\infty)}\|\mathbf{\Pi}_N\dot{\mathbf{u}}(t)\| = \gamma < \infty$. For any $\alpha,\beta \in \mathbb{R}_{>0}$, (S21) over $\mathcal{G}$ starting from $x^i(0) \in \mathbb{R}$ and $v^i(0) \in \mathbb{R}$ with $\Sigma_{j=1}^N v^j(0) = 0$, where each agent $i \in \{1,\dots,N\}$ communicates with its neighbors at times $\{t_k^i\}_{k\in\mathbb{N}} \subset \mathbb{R}_{\geq 0}$, starting at $t_1^i = 0$, determined by (S22) with $\epsilon \in \mathbb{R}_{>0}^N$, satisfies

$$\limsup_{t\to\infty}|x^i(t) - u^{\text{avg}}(t)| \leq \frac{\gamma + \beta\|\mathbf{L}\|\|\epsilon\|}{\beta\hat{\lambda}_2}, \qquad \text{(S24)}$$

for $i \in \{1,\dots,N\}$ with an exponential rate of convergence of $\min\{\alpha, \beta\hat{\lambda}_2\}$. Furthermore, the interexecution times of agent $i \in \{1,\dots,N\}$ are lower bounded by (S23). □

The expected tradeoff between the desire for longer interevent time and the adverse effect on systems convergence and performance is captured in (S23) and (S24). The lower bound τ^i in (S23) on the inter-event times allows a designer to compute bounds on the maximum number of communication rounds (and associated energy spent) by each agent $i \in \{1,\ldots,N\}$ (and hence the network) during any given time interval. It is interesting to analyze how this lower bound depends on the various problem ingredients: τ^i is an increasing function of ϵ_i and a decreasing function of α and c^i. Through the latter variable, the bound also depends on the graph topology and the design parameter β. Given the definition of c^i, we can deduce that the faster an input of an agent is changing (larger κ^i) or the farther the agent initially starts from the average of the inputs, the more often that agent would need to trigger communication. The connection between the network performance and the communication overhead can also be observed here. Increasing β or decreasing ϵ_i to improve the ultimate tracking error bound (S24) results in smaller inter-event times. Given that the rate of convergence of (S21) under (S22) is $\min\{\alpha, \beta\hat{\lambda}_2\}$, decreasing α to increase the inter-event times slows down the convergence.

When the interaction topology is a connected graph, the properties of the Laplacian enable the identification of an alternative event-triggered communication law that, compared to (S22), has a longer inter-event time but similar dynamic average tracking performance. Consider the sequence of communication times $\{t_k^i\}_{k\in\mathbb{N}}$ determined by

$$t_{k+1}^i = \operatorname{argmax}\{t \in [t_k^i, \infty) \,\big|\, |\hat{x}^i(t) - x^i(t)|^2 \le \frac{1}{4\mathrm{d}_{\mathrm{out}}^i}\sum_{j=1}^N \mathrm{a}_{ij}(t)\,|\hat{x}^i(t) - \hat{x}^j(t)|^2 + \frac{1}{4\mathrm{d}_{\mathrm{out}}^i}\epsilon_i^2\}. \tag{S25}$$

Compared to (S22), the extra term $1/(4\mathrm{d}_{\mathrm{out}}^i)\Sigma_{j=1}^N \mathrm{a}_{ij}(t)|\hat{x}^i(t) - \hat{x}^j(t)|^2$ in the communication law (S25) allows agents to have longer inter-event times. Formally, the interexecution times of agent $i \in \{1,\ldots,N\}$ implementing (S25) are lower bounded by

$$\tau^i = \frac{1}{\alpha}\ln\left(1 + \frac{\alpha\epsilon_i}{2\bar{c}^i\sqrt{\mathrm{d}_{\mathrm{out}}^i}}\right), \tag{S26}$$

for positive constants $\bar{c}^i$; see [S15] for explicit expressions. Numerical examples in [S15] show that the implementation of (S25) for connected graphs results in inter-event times longer than the ones of the event-triggered law (S22). Figure S2 shows one of those examples. Similar results can also be derived for time-varying, jointly connected graphs (see [S15] for a complete exposition).

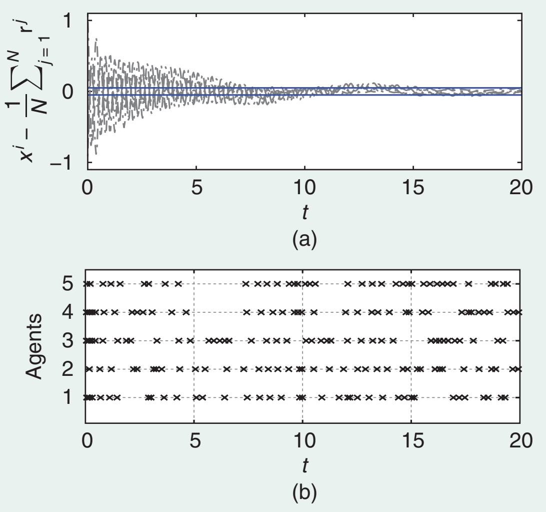


FIGURE S2 A comparison between the event-triggered algorithm (S21) employing the event-triggered communication law (S25) and the Euler discretized implementation of (25), as described in (S15) with fixed step size [S15]. Both of these algorithms use $\alpha = 1$ and $\beta = 4$. The network is a weight-balanced digraph of five agents with unit weights. The inputs are $r^1(t) = 0.5\sin(0.8t)$, $r^2(t) = 0.5\sin(0.7t) + 0.5\cos(0.6t)$, $r^3(t) = \sin(0.2t) + 1$, $r^4(t) = \operatorname{atan}(0.5t)$, and $r^5(t) = 0.1\cos(2t)$. In plot (a), the black (respectively, gray) lines correspond to the tracking error of the event-triggered algorithm (S21) employing event-triggered law (S25) with $\epsilon_i/(2\sqrt{\mathrm{d}_{\mathrm{out}}^i}) = 0.1$ [respectively, the Euler discretized algorithm (S15) with fixed step size $\delta = 0.12$]. Recall from "Euler Discretizations of Continuous-Time Dynamic Average Consensus Algorithms" that convergence for (S15) is guaranteed if $\delta \in (0, \min\{\alpha^{-1}, \beta^{-1}(\mathrm{d}_{\max}^{\mathrm{out}})^{-1}\})$, which, for this example, results in $\delta \in (0, 0.125)$. The horizontal blue lines show the ± 0.05 error bound for reference. Part (a) shows that both algorithms exhibit comparable tracking performance. Part (b) shows the communication times of each agent using the event-triggered strategy. The number of times that agents $\{1,\ldots,5\}$ communicate in the time interval $[0, 20]$ is $(39, 40, 42, 40, 39)$, respectively, when implementing event-triggered communication (S25). These numbers are significantly smaller than the communication rounds required by each agent in the Euler discretized algorithm (S15) $(20/0.12 \simeq 166$ rounds).

REFERENCES

[S22] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Hoboken, NJ: Wiley, 2005.

[S23] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Proc. IEEE Conf. Decision and Control*, 2012, pp. 3270–3285.

[S24] L. Hetel et al., "Recent developments on the stability of systems with aperiodic sampling: An overview," *Automatica*, vol. 76, pp. 309–335, Feb. 2017.

[S25] C. Nowzari, E. García, and J. Cortés. Event-triggered control and communication of network systems for multi-agent consensus. 2017. [Online]. Available: arXiv: 1712.00429

[S26] L. Ding, Q. L. Han, X. Ge, and X. M. Zhang, "An overview of recent advances in event-triggered consensus of multiagent systems," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1110–1123, 2018.

We believe that the interconnection of dynamic average consensus algorithms with other coordination layers in network systems is a fertile area for both research and applications. Dynamic average consensus algorithms are a versatile tool in interconnected scenarios where it is necessary to compute changing estimates of quantities that are employed by other coordination algorithms and whose execution in turn affects the time-varying signals available to the individual agents. This was illustrated in the "Applications of Dynamic Average Consensus in Network Systems" section, which described how (in resource allocation problems) a group of distributed energy resources can collectively estimate the mismatch between the aggregated power injection and the desired load using dynamic average consensus. The computed mismatch, in turn, informs the distributed energy resources in their decision-making process seeking to determine the power injections that optimize their generation cost, which, in turn, changes the mismatch computed by the dynamic average consensus algorithm. The fact that the time-varying nature of the signals is driven by a dynamic process that itself uses the output of the dynamic average consensus algorithms opens the way for the use of many concepts germane to systems and control, including stable interconnections, ISS, and passivity. Along these lines, we could also think of self-tuning mechanisms embedded within dynamic average consensus algorithmic solutions that tune the algorithm execution based on the evolution of the time-varying signals.

Another interesting topic for future research is the privacy preservation of the signals available to the agents in the dynamic average consensus problem. Protecting the privacy and confidentiality of data is a critical issue in emerging distributed automated systems deployed in a variety of scenarios, including power networks, smart transportation, the Internet of Things, and manufacturing systems. In such scenarios, the ability of a network system to optimize its operation, fuse information, compute common estimates of unknown quantities, and agree on a common worldview while protecting sensitive information is crucial. In this respect, the design of privacy-preserving dynamic average consensus algorithms is in its infancy. Interestingly, the dynamic nature of the problem might offer advantages in this regard with respect to the static average consensus problem. For instance, in differential privacy, where the designer makes it provably difficult for an adversary to make inferences about individual records from published outputs or even detect the presence of an individual in the data set, it is known that privacy guarantees weaken as more queries are made to the same database. However, if the database is changing, this limitation no longer applies, and this opens the way to studying how privacy guarantees change with the rate of variation of the time-varying signals in the dynamic average consensus problem.

ACKNOWLEDGMENTS

The work of S.S. Kia was supported by National Science Foundation awards ECCS-1653838 and IIS-SAS-1724331. The work of J. Cortés was supported by NSF award CNS-1446891 and Air Force Office of Scientific Research award FA9550-15-1-0108. The work of S. Martinez was supported by the Air Force Office of Scientific Research award FA9550-18-1-0158 and Defense Advanced Research Projects Agency (Lagrange) award N66001-18-2-4027.

AUTHOR INFORMATION

Solmaz S. Kia (solmaz@uci.edu) is an assistant professor of mechanical and aerospace engineering at the University of California, Irvine (UCI). She received the Ph.D. degree in mechanical and aerospace engineering from UCI in 2009 and the M.Sc. and B.Sc. degrees in aerospace engineering from the Sharif University of Technology, Tehran, Iran, in 2004 and 2001, respectively. She was a senior research engineer at SySense Inc., El Segundo, California, from June 2009 to September 2010. She held postdoctoral positions in the Department of Mechanical and Aerospace Engineering at the University of California, San Diego, and UCI. She was a recipient of the University of California President's Postdoctoral Fellowship from 2012 to 2014 and National Science Foundation CAREER Award in 2017. Her main research interests, in a broad sense, include distributed optimization/coordination/estimation, nonlinear control theory, and probabilistic robotics.

Bryan Van Scoy is a postdoctoral researcher at the University of Wisconsin–Madison. He received the Ph.D. degree in electrical engineering and computer science from Northwestern University, Evanston, Illinois, in 2017 and the B.S. and M.S. degrees in applied mathematics along with the B.S.E. in electrical engineering from the University of Akron, Ohio, in 2012. His research interests include distributed algorithms for multiagent systems and the analysis and design of optimization algorithms.

Jorge Cortés is a professor of mechanical and aerospace engineering at the University of California, San Diego. He received the Licenciatura degree in mathematics from the Universidad de Zaragoza, Spain, in 1997 and the Ph.D. degree in engineering mathematics from the Universidad Carlos III de Madrid, Spain, in 2001. He held postdoctoral positions with the University of Twente, The Netherlands, and the University of Illinois at Urbana-Champaign. He was an assistant professor with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, from 2004 to 2007. He is the author of *Geometric, Control and Numerical Aspects of Nonholonomic Systems* (Springer-Verlag, 2002) and coauthor (together with F. Bullo and S. Martínez) of *Distributed Control of Robotic Networks* (Princeton University Press, 2009). He was an IEEE Control Systems Society Distinguished Lecturer (2010–2014) and is an IEEE Fellow. His current research interests include cooperative control; network science; game theory; multiagent coordination in robotics, power systems, and neuroscience; geometric and distributed optimization; nonsmooth analysis; and geometric mechanics and control.

Randy A. Freeman is a professor of electrical engineering and computer science at Northwestern University, Evanston, Illinois. He received the Ph.D. degree in electrical engineering from the University of California, Santa Barbara, in 1995. He has been associate editor of *IEEE Transactions on Automatic Control*. His research interests include nonlinear systems, distributed control, multiagent systems, robust control, optimal control, and oscillator synchronization. He received the National Science Foundation CAREER Award in 1997. He has been a member of the IEEE Control System Society Conference Editorial Board since 1997 and has served on program and operating committees for the American Control Conference and the IEEE Conference on Decision and Control.

Kevin M. Lynch is a professor and chair of the Mechanical Engineering Department, Northwestern University, Evanston, Illinois. He received the B.S.E.E. degree in electrical engineering from Princeton University, New Jersey, in 1989 and the Ph.D. degree in robotics from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1996. He is a member of the Neuroscience and Robotics Laboratory and the Northwestern Institute on Complex Systems. His research interests include dynamics, motion planning, and control for robot manipulation and locomotion; self-organizing multiagent systems; and functional electrical stimulation for restoration of human function. He is a coauthor of the textbooks *Principles of Robot Motion* (MIT Press, 2005), *Embedded Computing and Mechatronics* (Elsevier, 2015), and *Modern Robotics: Mechanics, Planning, and Control* (Cambridge University Press, 2017).

Sonia Martínez is a professor of mechanical and aerospace engineering at the University of California, San Diego. She received the Ph.D. degree in engineering mathematics from the Universidad Carlos III de Madrid, Spain, in May 2002. Following a year as a visiting assistant professor of applied mathematics at the Technical University of Catalonia, Spain, she obtained a Postdoctoral Fulbright Fellowship and held appointments at the Coordinated Science Laboratory of the University of Illinois at Urbana-Champaign during 2004 and the Center for Control, Dynamical Systems and Computation of the University of California, Santa Barbara, during 2005. In a broad sense, her main research interests include the control of network systems, multiagent systems, nonlinear control theory, and robotics. For her work on the control of underactuated mechanical systems, she received the Best Student Paper Award at the 2002 IEEE Conference on Decision and Control. She was the recipient of a National Science Foundation CAREER Award in 2007. For the paper "Motion Coordination with Distributed Information," coauthored with Jorge Cortés and Francesco Bullo, she received the 2008 Control Systems Magazine Outstanding Paper Award. She has served on the editorial boards of *European Journal of Control* (2011–2013) and currently serves on the editorial board of *Journal of Geometric Mechanics* and *IEEE Transactions on Control of Network Systems*.

REFERENCES

[1] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, 2007.
[2] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control* (Communications and Control Engineering). New York: Springer-Verlag, 2008.
[3] W. Ren and Y. Cao, *Distributed Coordination of Multi-Agent Networks* (Communications and Control Engineering). New York: Springer-Verlag, 2011.
[4] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multi-vehicle cooperative control: Collective group behavior through local interaction," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 71–82, 2007.
[5] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, "Information weighted consensus filters and their application in distributed camera networks," *IEEE Trans. Autom. Control*, vol. 58, no. 12, pp. 3112–3125, 2013.
[6] D. Tian, J. Zhou, and Z. Sheng, "An adaptive fusion strategy for distributed information estimation over cooperative multi-agent networks," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3076–3091, 2017.
[7] P. Yang, R. Freeman, and K. Lynch, "Optimal information propagation in sensor networks," in *Proc. 2006 IEEE Int. Conf. Robotics and Automation*, pp. 3122–3127.
[8] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Trans. Robot.*, vol. 24, no. 3, pp. 710–724, 2008.
[9] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Trans. Autom. Control*, vol. 53, no. 11, pp. 2480–2496, 2008.
[10] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.
[11] R. Aragüés, J. Cortés, and C. Sagüés, "Distributed consensus on robot networks for dynamically merging feature-based maps," *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 840–854, 2012.
[12] S. Das and J. M. F. Moura, "Distributed Kalman filtering with dynamic observations consensus," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4458–4473, 2015.
[13] F. Chen and W. Ren, "A connection between dynamic region-following formation control and distributed average tracking," *IEEE Trans. Cybern.*, vol. 48, no. 6, pp. 1760–1772, 2018.
[14] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, 2004.
[15] W. Ren, "Multi-vehicle consensus with a time-varying reference state," *Syst. Control Lett.*, vol. 56, no. 2, pp. 474–483, 2007.
[16] K. D. Listmann, M. V. Masalawala, and J. Adamy, "Consensus for formation control of nonholonomic mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2009, pp. 3886–3891.
[17] M. Porfiri, G. D. Roberson, and D. J. Stilwell, "Tracking and formation control of multiple autonomous agents: A two-level consensus approach," *Automatica*, vol. 43, no. 8, pp. 1318–1328, 2007.
[18] S. Ghapani, S. Rahili, and W. Ren, "Distributed average tracking for second-order agents with nonlinear dynamics," in *Proc. American Control Conf.*, 2016, pp. 4636–4641.
[19] S. S. Kia, J. Cortés, and S. Martínez, "Dynamic average consensus under limited control authority and privacy requirements," *Int. J. Robust Nonlinear Control*, vol. 25, no. 13, pp. 1941–1966, 2015.
[20] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *Proc. IEEE Conf. Decision and Control*, 2005, pp. 8179–8184.
[21] R. Olfati-Saber, "Kalman-consensus filter: Optimality, stability, and performance," in *Proc. IEEE Conf. Decision and Control*, 2009, pp. 7036–7042.
[22] W. Qi, P. Zhang, and Z. Deng, "Robust sequential covariance intersection fusion Kalman filtering over multi-agent sensor networks with measurement delays and uncertain noise variances," *Acta Autom. Sin.*, vol. 40, no. 11, pp. 2632–2642, 2014.
[23] G. Wang, N. Li, and Y. Zhang, "Diffusion distributed Kalman filter over sensor networks without exchanging raw measurements," *Signal Process.*, vol. 132, pp. 1–7, Mar. 2017.
[24] R. Aragues, C. Sagues, and Y. Mezouar, "Feature-based map merging with dynamic consensus on information increments," *Autonomous Robots*, vol. 38, no. 3, pp. 243–259, 2015.
[25] W. Ren and U. M. Al-Saggaf, "Distributed Kalman–Bucy filter with embedded dynamic averaging algorithm," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1722–1730, 2018.

[26] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009.
[27] B. Johansson, M. Rabi, and M. Johansson, “A randomized incremental subgradient method for distributed optimization in networked systems,” *SIAM J. Control Optim.*, vol. 20, no. 3, pp. 1157–1170, 2009.
[28] J. Wang and N. Elia, “A control perspective for centralized and distributed convex optimization,” in *Proc. IEEE Conf. Decision and Control*, 2011, pp. 3800–3805.
[29] M. Zhu and S. Martínez, “On distributed convex optimization under inequality and equality constraints,” *IEEE Trans. Autom. Control*, vol. 57, no. 1, pp. 151–164, 2012.
[30] J. Lu and C. Y. Tang, “Zero-gradient-sum algorithms for distributed convex optimization: The continuous-time case,” *IEEE Trans. Autom. Control*, vol. 57, no. 9, pp. 2348–2354, 2012.
[31] B. Gharesifard and J. Cortés, “Distributed continuous-time convex optimization on weight-balanced digraphs,” *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 781–786, 2014.
[32] S. S. Kia, J. Cortés, and S. Martínez, “Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication,” *Automatica*, vol. 55, pp. 254–264, May 2015.
[33] G. Qu and N. Li, “Accelerated distributed Nesterov gradient descent for convex and smooth functions,” in *Proc. IEEE Conf. Decision and Control*, 2017, pp. 2260–2267.
[34] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. New York: Springer-Verlag, 2014.
[35] R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, “Analysis of Newton–Raphson consensus for multi-agent convex optimization under asynchronous and lossy communication,” in *Proc. IEEE Conf. Decision and Control*, 2015, pp. 418–424.
[36] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, “Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes,” in *Proc. IEEE Conf. Decision and Control*, 2015, pp. 2055–2060.
[37] D. Varagnolo, F. Zanella, P. G. A. Cenedese, and L. Schenato, “Newton–Raphson consensus for distributed convex optimization,” *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 994–1009, 2016.
[38] P. D. Lorenzo and G. Scutari, “NEXT: In-network nonconvex optimization,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 120–136, 2016.
[39] A. Nedic, A. Olshevsky, and W. Shi, “Achieving geometric convergence for distributed optimization over time-varying graphs,” *SIAM J. Optim.*, vol. 27, no. 4, pp. 2597–2633, 2017.
[40] A. J. Wood, F. Wollenberg, and G. B. Sheble, *Power Generation, Operation and Control*, 3rd ed. Hoboken, NJ: Wiley, 2013.
[41] A. Cherukuri and J. Cortés, “Distributed generator coordination for initialization and anytime optimization in economic dispatch,” *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 3, pp. 226–237, 2015.
[42] R. Madan and S. Lall, “Distributed algorithms for maximum lifetime routing in wireless sensor networks,” *IEEE Trans. Wireless Commun.*, vol. 5, no. 8, pp. 2185–2193, 2006.
[43] G. M. Heal, “Planning without prices,” *Rev. Economic Stud.*, vol. 36, no. 3, pp. 347–362, 1969.
[44] K. J. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Linear and Nonlinear Programming*. Stanford, CA: Stanford Univ. Press, 1958.
[45] A. Cherukuri, B. Gharesifard, and J. Cortés, “Saddle-point dynamics: Conditions for asymptotic stability of saddle points,” *SIAM J. Control Optim.*, vol. 55, no. 1, pp. 486–511, 2017.
[46] A. Cherukuri and J. Cortés, “Initialization-free distributed coordination for economic dispatch under varying loads and generator commitment,” *Automatica*, vol. 74, pp. 183–193, Dec. 2016.
[47] S. S. Kia, “Distributed optimal in-network resource allocation algorithm design via a control theoretic approach,” *Syst. Control Lett.*, vol. 107, pp. 49–57, Sept. 2017.
[48] J. Tsitsiklis, “Problems in decentralized decision making and computation,” Ph.D. dissertation, MIT, Cambridge, MA, 1984.
[49] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, “Dynamic consensus for mobile networks,” in *Proc. IFAC World Congr.*, 2005, Art. no. Mo-A09-TO/5.
[50] M. Fiedler, “Algebraic connectivity of graphs,” *Czechoslovak. Math. J.*, vol. 23, no. 2, pp. 298–305, 1973.
[51] N. M. M. de Abreu, “Old and new results on algebraic connectivity of graphs,” *Linear Algebra Its Applicat.*, vol. 423, no. 1, pp. 53–73, 2007.
[52] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri, “Communication constraints in the average consensus problem,” *Automatica*, vol. 44, no. 3, pp. 671–684, 2008.
[53] R. A. Freeman, P. Yang, and K. M. Lynch, “Stability and convergence properties of dynamic average consensus estimators,” in *Proc. IEEE Conf. Decision and Control*, 2006, pp. 398–403.
[54] S. S. Kia, J. Cortés, and S. Martínez, “Singularly perturbed filters for dynamic average consensus,” in *Proc. European Control Conf.*, 2013, pp. 1758–1763.
[55] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 2002.
[56] B. Van Scoy, R. A. Freeman, and K. M. Lynch, “Optimal worst-case dynamic average consensus,” in *Proc. American Control Conf.*, 2015, pp. 5324–5329.
[57] B. Van Scoy, R. A. Freeman, and K. M. Lynch, “Design of robust dynamic average consensus estimators,” in *Proc. IEEE Conf. Decision and Control*, 2015, pp. 6269–6275.
[58] B. Van Scoy, R. A. Freeman, and K. M. Lynch, “Exploiting memory in dynamic average consensus,” in *Proc. 53rd Annu. Allerton Conf. Communication, Control, and Computing*, 2015, pp. 258–265.
[59] B. N. Oreshkin, M. J. Coates, and M. G. Rabbat, “Optimization and analysis of distributed averaging with short node memory,” *IEEE Trans. Signal Process.*, vol. 58, no. 5, pp. 2850–2865, 2010.
[60] T. Erseghe, D. Zennaro, E. Dall’Anese, and L. Vangelista, “Fast consensus by the alternating direction multipliers method,” *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5523–5537, 2011.
[61] E. Kokiopoulou and P. Frossard, “Polynomial filtering for fast convergence in distributed consensus,” *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 342–354, 2009.
[62] Y. Yuan, J. Liu, R. M. Murray, and J. Gonçalves, “Decentralised minimal-time dynamic consensus,” in *Proc. American Control Conf.*, 2012, pp. 800–805.
[63] E. Montijano, J. I. Montijano, and C. Sagüés, “Chebyshev polynomials in distributed consensus applications,” *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 693–706, 2013.
[64] M. L. Elwin, R. A. Freeman, and K. M. Lynch, “A systematic design process for internal model average consensus estimators,” in *Proc. IEEE Conf. Decision and Control*, 2013, pp. 5878–5883.
[65] M. L. Elwin, R. A. Freeman, and K. M. Lynch, “Worst-case optimal average consensus estimators for robot swarms,” in *Proc. 2014 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014, pp. 3814–3819.
[66] H. Bai, R. A. Freeman, and K. M. Lynch, “Robust dynamic average consensus of time-varying inputs,” in *Proc. IEEE Conf. Decision and Control*, 2010, pp. 3104–3109.
[67] H. Bai, “Adaptive motion coordination with an unknown reference velocity,” in *Proc. American Control Conf.*, 2015, pp. 5581–5586.
[68] B. Van Scoy, R. A. Freeman, and K. M. Lynch, “Feedforward estimators for the distributed average tracking of bandlimited signals in discrete time with switching graph topology,” in *Proc. IEEE Conf. Decision and Control*, 2016, pp. 4284–4289.
[69] F. Chen, Y. Cao, and W. Ren, “Distributed average tracking of multiple time-varying reference signals with bounded derivatives,” *IEEE Trans. Autom. Control*, vol. 57, no. 12, pp. 3169–3174, 2012.
[70] J. George, R. A. Freeman, and K. M. Lynch, “Robust dynamic average consensus algorithm for signals with bounded derivatives,” in *Proc. American Control Conf.*, 2017, pp. 352–357.
[71] S. Rahili and W. Ren, “Heterogeneous distributed average tracking using nonsmooth algorithms,” in *Proc. American Control Conf.*, 2017, pp. 691–696.
[72] L. Yu, J. P. Barbot, D. Benmerzouk, D. Boutat, T. Floquet, and G. Zheng, *Discussion About Sliding Mode Algorithms, Zeno Phenomena and Observability*. New York: Springer-Verlag, 2012, pp. 199–219.
[73] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice Hall, 1991.
[74] L. Fridman and A. Levant, *Higher Order Sliding Modes*. Boca Raton, FL: CRC, 2002, pp. 53–101.
[75] E. G. Rojo-Rodriguez, E. J. Ollervides, J. G. Rodriguez, E. S. Espinoza, P. Zambrano-Robledo, and O. Garcia, “Implementation of a super twisting controller for distributed formation flight of multi-agent systems based on consensus algorithms,” in *Proc. Int. Conf. Unmanned Aircraft Systems*, Miami, FL, 2017, pp. 1101–1107.
[76] M. Zhu and S. Martínez, “Discrete-time dynamic average consensus,” *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.
[77] E. Montijano, J. I. Montijano, C. Sagüés, and S. Martínez, “Step size analysis in discrete-time dynamic average consensus,” in *Proc. American Control Conf.*, 2014, pp. 5127–5132.