Tutorial on Dynamic Average Consensus

The problem, its applications, and the algorithms

S. S. Kia B. Van Scoy J. Cortés R. A. Freeman K. M. Lynch S. Martínez

The need to agree on linear combinations of dynamically changing local parameters or signals emerges in many applications in multi-agent systems and sensor networks. This dynamic agreement problem can be cast as a dynamic average consensus problem. Simply put, the problem is for a group of agents to cooperate in order to track the average of locally available time-varying reference signals, where each agent is only capable of local computations and communicating with local neighbors, see Figure 1. It is likely that the reader is familiar with the static version of this problem, which we term here the static average consensus problem, where agents seek to agree on a specific combination of fixed quantities [1, 2]. 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. Our primary intention, rather than providing a full account of all the available literature, is to introduce the reader, in a tutorial fashion, to the main ideas behind dynamic average consensus algorithms, the performance trade-offs considered in their design, and the requirements needed for their analysis and convergence guarantees.

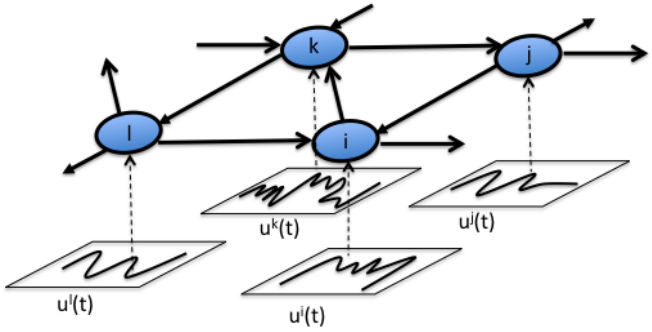


Figure 1: A group of communication agents, each endowed with a time-varying reference signal.

What is Dynamic Average Consensus?

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 "Sidebar 1: Preliminaries on Graph Theory" for further details and the graph related notations used throughout the article. 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

$$\text{CT:} \quad \mathsf{u}^{\text{avg}}(t) := \frac{1}{N} \sum\nolimits_{i=1}^{N} \mathsf{u}^i(t)$$
$$\text{DT:} \quad \mathsf{u}^{\text{avg}}(k) := \frac{1}{N} \sum\nolimits_{i=1}^{N} \mathsf{u}^i(k)$$

in continuous time (CT) and discrete time (DT), respectively. For discrete-time signals and algorithms, for any variable p sampled at time t_k, we use the shorthand notation $p(k)$ or p_k to refer to $p(t_k)$.

For reasons that we specify below, we are specifically interested in the design of distributed algorithms, meaning that to obtain the average, the policy that each agent implements only depends on its variables (represented by J^i, which include its own reference signal) and those of its out-neighbors (represented by $\{I^j\}_{j \in \mathcal{N}^i_{\text{out}}}$).

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

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

with proper initialization if necessary, we have $x^i(t) \to \mathsf{u}^{\text{avg}}(t)$ 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, we are making the convention 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, we seek a *driving command* for each agent $i \in \{1, \dots, N\}$ so that

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

under proper initialization if necessary, accomplishes $x^i(t_k) \to \mathsf{u}^{\text{avg}}(k)$ as $t_k \to \infty$.

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

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

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^j-th transmission time of agent j, which is not necessarily synchronous with the transmission time of other agents in the network.

The consideration of simple dynamics of the form in (1), (2) and (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, position of a mobile agent in a robotic team. In such cases, one can leverage the discussion here by, for instance, having agents compute reference signals that are to be tracked by the states with more complex dynamics. Interested readers can consult "Sidebar 2: Further Reading" for a list of relevant literature on dynamic average consensus problems for higher-order dynamics.

Centralized solutions

The difficulty in the dynamic average consensus problem is that the information is distributed across the network. An analytically straightforward solution to the dynamic average consensus problem then is to get all of the information in a single place, do the computation (in other words, calculate the average), and then send the solution back through the network to each agent. This is known as a *centralized* solution.

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 since 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 which 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 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, 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 identified above. Also, although this approach

works reasonably well for small size networks, its communication and storage costs scale poorly in terms of the network size, incurring in costs of order $O(N)$ per agent per timestep. Additionally, flooding requires an accurate bookkeeping effort from each agent.

Desirable properties in a dynamic average consensus algorithm

Given the drawbacks of centralized solutions, here we identify a number of desirable properties when designing algorithmic solutions to the dynamic average consensus problem. The algorithm should be

- *scalable*, so that the amount of computations and resources required on each agent does not grow with the network size,
- *robust* to the disturbances present in practical scenarios, such as communication delays and packet drops, agents entering/leaving the network, noisy measurements, and
- *correct*, meaning that the algorithm converges to the exact average or, alternatively, a formal guarantee can be given about how far from the exact average it will be.

Regarding the last property, to achieve agreement, network connectivity should be such that information about the local reference input of each agent reaches other agents frequently enough. Also, as the information of each agent takes some time to propagate through the network, we can expect 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. Therefore, a recurring theme throughout the article will be how the convergence guarantees of dynamic average consensus algorithms depend on the network connectivity and on the rate of change of the reference signal of each agent.

Why do We need Specialized Algorithms for Dynamic Average Consensus?

As mentioned above, static average consensus refers to the problem when the reference signals of the agents do not change with time. The static problem has been extensively studied in the literature [1, 2, 3, 4], and several simple and efficient distributed algorithms exist with exact convergence guarantees. Given this, a natural approach to deal with the distributed solution of the dynamic average consensus problem is the following: zero-order sample the reference signals and successively use a static average consensus algorithm between sampling times. If this approach were successful, then it would mean that we do not need to worry about designing specific algorithms to solve the dynamic average consensus problem and we can simply rely on the algorithmic solutions available for static average consensus.

As the reader might have correctly guessed already, this is not the case. In order for this idea to work, one would essentially need a static average consensus algorithm which 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 will operate with some error, whose size depends on its speed of convergence and how fast inputs change. To illustrate this point better, we use a simple 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 6 agents with the communication topology of directed ring monitors this process by taking synchronous samples, each according to

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

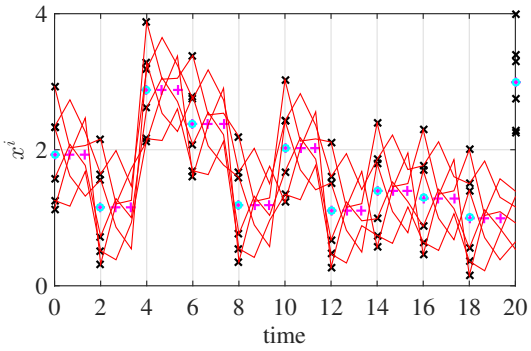
where a^i and b^i are fixed unknown error sources in the measurement of agent $i \in \{1, \ldots, 6\}$. To reduce the effect of measurement errors, after each sampling, every agent wants to obtain the average of the measurements across the network before the next sampling time. For our numerical simulations,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

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

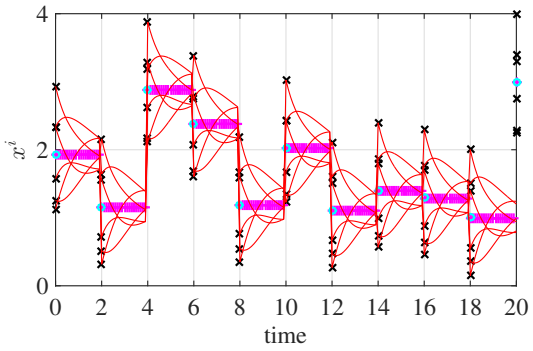
by the current sampled reference values, $x^i(0) = \mathsf{u}^i(m)$, and implements it with an admissible timestep δ until just before the next sampling time $m+1$; (b) at time $m = 0$, agents start executing a dynamic average consensus algorithm (more specifically, the strategy (S13) which is described 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 here k is the communication time index. Figure 2 compares the tracking performance of these two approaches. One can observe 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 of specifically designed distributed algorithms that take into account the particular features of the dynamic average consensus problem.

Applications of Dynamic Average Consensus in Network Systems

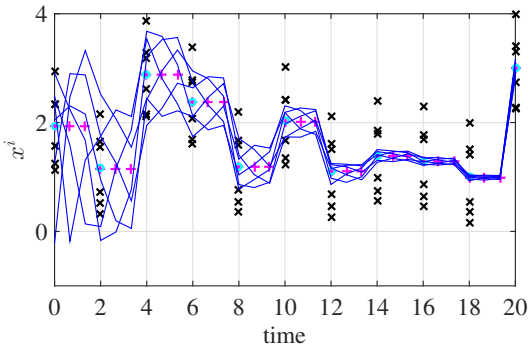
The ability to compute the average of a set of time-varying reference signals turns out to be useful in numerous applications, and this explains why distributed algorithmic solutions have found their way into many seemingly different problems involving the interconnection of dynamical systems. This section provides a selected overview of problems to further motivate the reader to learn about



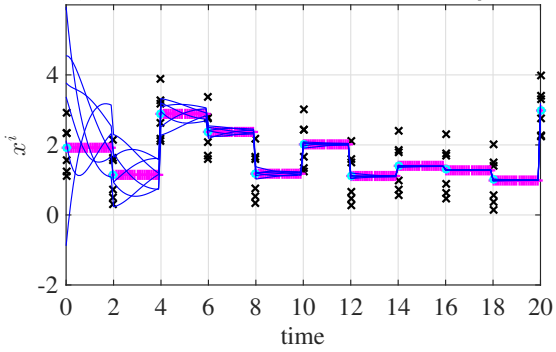
(a) Static algorithm; 3 communications in $t \in [m, m+1]$



(b) Static algorithm; 20 communications in $t \in [m, m+1]$



(c) Dynamic algorithm; 3 communications in $t \in [m, m+1]$



(d) Dynamic algorithm; 20 communications in $t \in [m, m+1]$

Figure 2: Comparison of performance between a static average consensus algorithm re-initialized at each sampling time vs. a dynamic average consensus algorithm; The solid lines: red curves (resp. blue curves) represent the time history of the agreement state of each agent generated by the Laplacian static average consensus approach (resp. the dynamic average consensus of (S13)); $\times$: sampling points at $m\,\Delta t$; $\circ$: the average at $m\,\Delta t$; $+$: the average of reference signals at $k\,\delta$. The dynamic consensus algorithm tracks very closely the average as time goes by while the static consensus does not have enough time between sampling times to converge. This trend is preserved even if we increase the frequency of the communication between the agents. In these simulations we used $\alpha = \beta = 1$ in (S13).

dynamic average consensus algorithms and illustrate their range of applicability. Other applications of dynamic average consensus can be found in [5, 6, 7, 8, 9, 10, 11].

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. Often the tasks accomplished by mobile agents require dynamic motion coordination and formation among team members. Consensus algorithms have been commonly used in the design of formation control strategies [12, 13, 14]. Consensus algorithms are mainly used to arrive at agreement on the geometric center of formation, so that the formation can be achieved by spreading the agents in the

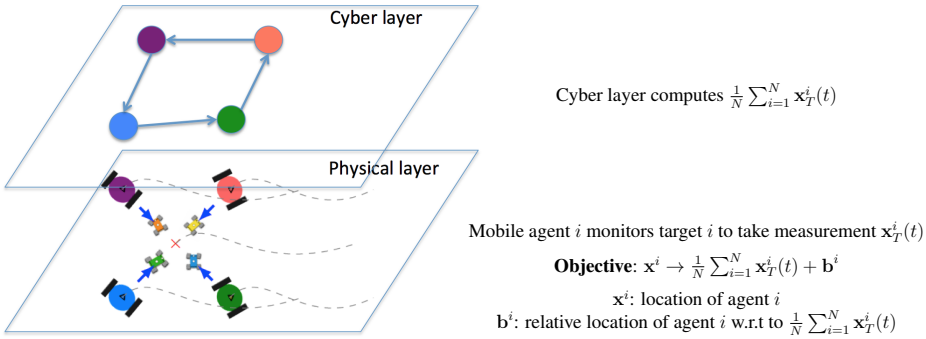


Figure 3: A two-layer consensus-based formation for tracking a team of mobile targets: the larger robots are the mobile agents and while the smaller robots are the mobile moving targets.

desired formation geometry about this geometric center, see [1]. However, most of the existing results are for static formation. Dynamic average consensus algorithms can effectively be used in dynamic formation control, where the geometric center of formation changes with time. Figure 3 depicts an example scenario in which a group of mobile agents track a team of mobile targets, with each agent monitoring a mobile target with location $\mathbf{x}_T^i$. The objective here is for the agents to follow the mobile target team by spreading out in a pre-specified formation consisted of each agent moving in $\mathbf{b}^i$ relative location 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 example 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 $\mathbf{b}^i$. Examples of the use of dynamic consensus algorithms in this two-layer approach with multi-agent systems with first-order, second-order or higher-order dynamics can be found in [15, 16, 17].

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

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

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 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 follows

- 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:

$$\begin{aligned}
&\mathbf{Y}^i(k+1) = \mathbf{H}^i(k+1)^\top \mathbf{R}^i(k+1)^{-1}\mathbf{H}^i(k+1),\\
&\mathbf{y}^i(k+1) = \mathbf{H}^i(k+1)^\top \mathbf{R}^i(k+1)^{-1}\mathbf{z}^i(k+1),\\
&\mathbf{P}^+(k+1) = (\mathbf{P}^-(k+1) - \sum\nolimits_{i=1}^{N} \mathbf{Y}^i(k+1))^{-1},\\
&\hat{\mathbf{x}}^+(k+1) = \hat{\mathbf{x}}^-(k+1) - \big(\sum\nolimits_{i=1}^{N} \mathbf{Y}^i(k+1) - \sum\nolimits_{i=1}^{N} \mathbf{y}^i(k+1)\,\hat{\mathbf{x}}^-(k+1)\big).
\end{aligned}$$

Despite its optimality, this implementation is not desirable in many sensor network applications due to 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 gained interest in recent years [18, 19, 20, 21, 22] 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 works [18, 23] employ 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 $\frac{1}{N}\sum_{i=1}^{N} \mathbf{y}^i(k+1)$ and $\frac{1}{N}\sum_{i=1}^{N} \mathbf{Y}^i(k+1)$. If agents know the size of the network, they can duplicate the update equation locally.

Distributed unconstrained convex optimization

The control literature has introduced in recent years a number of distributed algorithmic solutions [24, 25, 26, 27, 28, 29, 30, 31] 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 $f^i : \mathbb{R}^n \to \mathbb{R}$, $i \in \{1, \ldots, N\}$, seek to determine the minimizer of the joint global optimization problem

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

by local interactions with their neighboring agents. This problem appears in networked system applications such as multi-agent 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 [32]

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

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

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

where $\mathbf{x}(0), \mathbf{y}(0), \mathbf{v}(0) \in \mathbb{R}^n$, and $\{\alpha_k\}_{k=0}^{\infty}$ is 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, differntiable 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 $0 < \eta < \frac{1}{L}$ [32, Theorem 2.2.1 and Lemma 2.2.4]. The source of coupling in (6) is the cumulative gradient term $\frac{1}{N}\sum_{i=1}^{N} \nabla f^i(\mathbf{y}(k))$. The distributed solution developed in [31] to solve the optimization problem (5) over connected graphs uses a dynamic average consensus algorithm to compute this coupling term. Specifically, in the distributed algorithm of [31], each agent uses a local state $\mathbf{x}^i$, $\mathbf{y}^i$ and $\mathbf{v}^i$ to evolve (6) locally and employs the dynamic average algorithm of the form (15) below with input $\mathbf{u}^i(k) = \nabla f^i(\mathbf{y}^i(k))$ to estimate the gradient coupling term.

Distributed resource allocation

In optimal resource allocation, a group of agents work cooperatively to meet a demand in an efficient way. Each agent incurs a cost for the resources it provides. Let the cost function $f^i : \mathbb{R} \to \mathbb{R}$ of each agent $i \in \{1,\dots,N\}$ be convex and differentiable. The objective is to meet the demand $\mathsf{b} \in \mathbb{R}$ 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 i^{th} element of $\mathbf{x}^\star$ given by

$$\begin{aligned} \mathbf{x}^\star = &\arg\min_{\mathbf{x}\in\mathbb{R}^N} \sum\nolimits_{i=1}^{N} f^i(x^i), \ \text{ subject to} \\ &x^1 + \cdots + x^N - \mathsf{b} = 0, \end{aligned}$$

This problem appears in many optimal decision making tasks such as optimal dispatch in power networks [33, 34], optimal routing [35], and economic systems [36]. A centralized algorithmic solution is given by the popular saddle-point or primal-dual dynamics [37, 38] associated to the optimization problem,

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

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

If the local cost functions are strictly convex, 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 which appears in the right hand side of (7a). One can, however, employ dynamic average consensus to estimate this quantity online and feed it back into the algorithm. This is the approach taken in [39, 40]. This can be accomplished, for instance, by having agent i use the reference signal $x^i(t) - b/N$ (this assumes every agent knows the demand and the 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

It is not surprising that the initial synthesis of dynamic average consensus algorithms emerged from a careful look at static average consensus. In this section, we provide a brief review of standard algorithms for the static average consensus, and then build on this discussion to introduce the first dynamic average consensus algorithm presented in the paper.

Static average consensus algorithms

Consensus algorithms to solve the static average consensus problem have been studied at least as far back as [41]. 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 following form

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

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

for $i \in \{1, \dots, N\}$, 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 "Sidebar 1: Basic Notions from Graph Theory"). By stacking the agent variables into vectors, the static average consensus algorithms can be written compactly using the graph Laplacian as

$$\text{CT:} \qquad \dot{\mathbf{x}}(t) = -\mathbf{L}\,\mathbf{x}(t), \tag{9a}$$

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

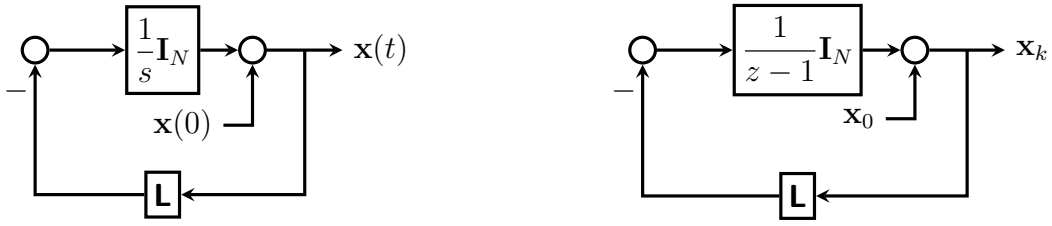
with $\mathbf{x}(0) = \mathsf{u}$. Here, note that the Laplacian matrices in the CT and DT algorithms are not the same, but a scaled version of each other (with the Laplacian matrix in the DT case incorporating the discretization stepsize).

When the communication graph is fixed, this is a linear time-invariant system and can be analyzed using standard time domain and frequency-domain techniques in control. Specifically, the frequency-domain representation of the dynamic average consensus algorithm output signal is given by

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

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

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 a static signals we have $\mathbf{U}(s) = \mathbf{u}$ and $\mathbf{U}(z) = \mathbf{u}$.



(a) Continuous time (b) Discrete time

Figure 4: 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).

The block diagram of these static average consensus algorithms is shown in Figure 4. 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, it can be shown that the error of these algorithms converges to zero. This is summarized next.

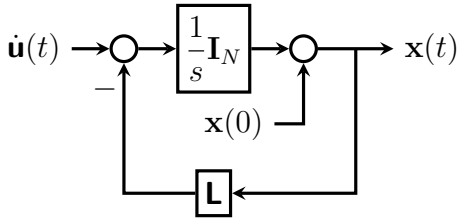
Theorem 0.1 (Convergence guarantees of the CT and DT static average consensus algorithms (8) [1])**.** *Suppose that the communication graph is constant, strongly connected, and weight-balanced digraph, and that the reference signals* u^i *at each agent* $i \in \{1, \dots, N\}$ *is a constant scalar. Then the following convergence results hold for the CT and DT static average consensus algorithms* (8)

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

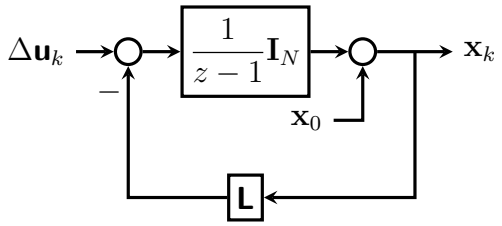
DT: *As* $k \to \infty$ *every agreement state* x_k^i*,* $i \in \{1, \dots, N\}$ *of the DT static average consensus algorithm* (8b) *converges to* u^{avg} *with an exponential rate no worse than* $\rho \in (0, 1)$*, provided that the Laplacian matrix satisfies* $\rho = \|\mathbf{I}_N - \mathbf{L} -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

A first design for dynamic average consensus

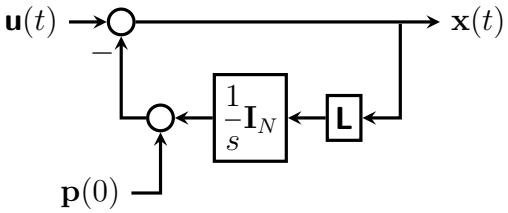
Since the reference signals enter the static average consensus algorithms (8) as initial conditions, they cannot track time-varying signals. Looking at the frequency-domain representation in Figure 4 of the static average consensus algorithms (8), it is clear that what is needed instead is to continuously 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 re-initialization. This is the basic observation made in [42], resulting in the systems shown in Figure 5.



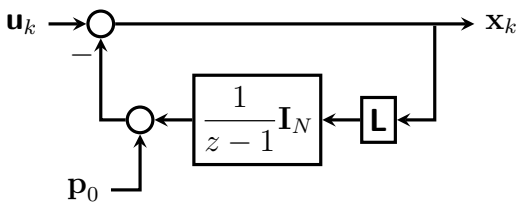
(a) Continuous-time algorithm (11)



(b) Discrete-time algorithm (15) where $\Delta\mathbf{u}_k := \mathbf{u}_{k+1} - \mathbf{u}_k$



(c) Equivalent continuous-time algorithm (12)



(d) Equivalent discrete-time algorithm (16)

Figure 5: Block diagram of the dynamic average consensus algorithms (11) and (16). Here, the reference signals are applied as inputs to the system. The feedback term is the integral of the error multiplied by the graph Laplacian. The top two systems are in the form (3) and explicitly require the derivative of the reference signals, while the bottom two (equivalent) systems do not require differentiating the reference signals.

More precisely, [42] argues that considering the static inputs as a dynamic step function, the algorithm

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

in which the reference value of the agents enter the dynamics as an external input, results in the same frequency representation (10a) (here $\mathsf{h}(t)$ is the Heaviside step function). Therefore, convergence to the average of reference values is guaranteed. Based on this observation, [42] proposes

one of the earliest algorithms for dynamic average consensus,

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

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

Algorithm (11) requires the knowledge of the derivative of the reference signals. In applications where the input signals are measured online, computing the derivative can be costly or error prone. A simple change of variables $p^i = \mathsf{u}^i - x^i$, $i \in \{1, \dots, N\}$, allows to write (11) in the equivalent form,

$$\dot{p}^i(t) = \sum_{j=1}^{N} a_{ij}(x^i(t) - x^j(t)), \quad p^i(0) = 0, \quad i \in \{1, \dots, N\}, \tag{12a}$$

$$x^i(t) = \mathsf{u}^i(t) - p^i(t). \tag{12b}$$

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). The convergence guarantees of this dynamics are as follows.

Theorem 0.2 (Convergence of (11) over connected graphs for dynamic input signals [42])**.** *Let $\mathcal{G}$ be a connected undirected graph. Consider the LTI system described by* (11)*. Suppose the input signal* $\mathbf{u}$ *has a Laplace transform with all poles in the left half-plane and at most one zero pole. Then,*

$$\lim_{t\to\infty} \left| x^i(t) - \mathsf{u}^{\text{avg}}(t) \right| = 0, \quad i \in \{1, \dots, N\}.$$

The time-domain analysis of the algorithm (11) reveals further information about its ultimate tracking response. Define the tracking error of agent i by

$$e^i(t) = x^i(t) - \mathsf{u}^{\text{avg}}(t).$$

To analyze the system, the error is decomposed into the *consensus direction* (that is, the direction $\mathbf{1}_N$) and the *disagreement directions* (that is, the directions orthogonal to $\mathbf{1}_N$). To this end, define the transformation matrix $\mathbf{T} = \begin{bmatrix} \frac{1}{\sqrt{N}}\mathbf{1}_N & \mathbf{R} \end{bmatrix}$ 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

$$\bar{\mathbf{e}} = \begin{bmatrix} \bar{e}_1 \\ \bar{\mathbf{e}}_{2:N} \end{bmatrix} = \mathbf{T}^\top \mathbf{e}. \tag{13}$$

In the new coordinates, (11) takes the form

$$\dot{\bar{e}}_1 = 0, \qquad \bar{e}_1(0) = \frac{1}{\sqrt{N}}\mathbf{1}_N^\top(\mathbf{x}(0) - \mathbf{u}(0)) = 0,$$

$$\dot{\bar{\mathbf{e}}}_{2:N} = -\mathbf{R}^\top \mathbf{L}\, \mathbf{R}\, \bar{\mathbf{e}}_{2:N} + \mathbf{R}^\top \dot{\mathbf{u}}, \qquad \bar{\mathbf{e}}_{2:N}(0) = \mathbf{R}^\top \mathbf{x}(0) = \mathbf{R}^\top \mathbf{u}(0).$$

Using the ISS bound on the trajectories of LTI systems (see "Sidebar 3: Input-to-State Stability of LTI Systems") one sees that the tracking error of each agent $i \in \{1, \dots, N\}$ while implementing (11) over a strongly connected and weight-balanced digraph is

$$\begin{aligned} |e^i(t)| &\leq \sqrt{\|\bar{\mathbf{e}}_{2:N}(t)\|^2 + |\bar{e}_1(t)|^2} \\ &\leq \mathrm{e}^{-\hat{\lambda}_2 t}\|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\mathbf{u}(0)\| + \frac{\sup_{0\leq\tau\leq t}\|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}(\tau)\|}{\hat{\lambda}_2}, \end{aligned} \tag{14}$$

for all $t \in \mathbb{R}_{\geq 0}$, where $\hat{\lambda}_2$ is the smallest non-zero eigenvalue of $\mathrm{Sym}(\mathbf{L})$. The tracking error bound (14) reveals the following interesting facts. First, it shows that the algorithm (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 because the ISS tracking bound due to external input depends on $\|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}(\tau)\|$ rather than $\|\dot{\mathbf{u}}(\tau)\|$. Note that if the reference signal of each agent $i \in \{1, \dots, N\}$ 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, we obtain $\|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}(\tau)\| = \|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)(\dot{\underline{\mathsf{u}}}(t)\mathbf{1}_N + \dot{\hat{\mathsf{u}}}(t))\| = \|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\dot{\hat{\mathsf{u}}}(t)\|$. This goes to show that the algorithm (11) properly uses the local knowledge about 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. Second, the tracking error bound (14) shows that as long as the uncommon part of reference signals has bounded rate, the algorithm (11) will track the average with some bounded error. Finally, (14) highlights the necessity for the special initialization (11b). Without it, a fixed off-set from perfect tracking will be present regardless of the type of reference input signals–one would expect that a proper dynamic consensus algorithm should be capable of perfectly tracking static reference signals.

One can also propose a discrete-time counterpart of the dynamic average consensus continuous-time algorithm (11) as

$$x_{k+1}^i = \Delta\mathsf{u}_k - x_k^i - \sum_{j=1}^N a_{ij}(x_k^i - x_k^j), \qquad x_0^i \in \mathbb{R}, \qquad i \in \{1, \dots, N\} \tag{15}$$

with $\Delta\mathsf{u}_k := \mathsf{u}_{k+1} - \mathsf{u}_k$, or equivalently,

$$p_{k+1}^i = p_k^i + \sum_{j=1}^N a_{ij}(x_k^i - x_k^j), \quad p_0^i \in \mathbb{R}, \quad i \in \{1, \dots, N\}, \tag{16a}$$

$$x_k^i = \mathsf{u}_k^i - p_k^i. \tag{16b}$$

The introduction of these algorithms has prepared us for a more in-depth treatment in our discussion on the design of dynamic average consensus algorithms. This is what we tackle next, where we discuss some of the shortcomings of the algorithms (11) and (16) regarding the requirement for correct initialization (the steady-state error depends on the initial condition $\mathbf{x}(0)$ or $\mathbf{x}_0$), having

Table 1: Arguments of the driving command in (1) for the reviewed continuous-time dynamic average consensus algorithms together with their initialization requirements.

Algorithm	(11)	(17)	(22)	(23)
$J^i(t)$	$\{x^i(t), \dot{\mathsf{u}}(t)\}$	$\{x^i(t), v^i(t), \mathsf{u}(t)\}$	$\{x^i(t), z^i(t), v^i(t), \mathsf{u}(t), \dot{\mathsf{u}}(t)\}$	$\{x^i(t), v^i(t), \mathsf{u}(t), \dot{\mathsf{u}}(t)\}$
$\{I^j(t)\}_{j\in\mathcal{N}^i_{\text{out}}}$	$\{x^j(t)\}_{j\in\mathcal{N}^i_{\text{out}}}$	$\{x^j(t), v^j(t)\}_{j\in\mathcal{N}^i_{\text{out}}}$	$\{z^j(t), v^j(t)\}_{j\in\mathcal{N}^i_{\text{out}}}$	$\{v^i(t)\}_{j\in\mathcal{N}^i_{\text{out}}}$
Initialization Requirement	$x^i(0) = \mathsf{u}^i(0)$	none	none	$\sum_{j=1}^N v^j(0) = 0$

slow convergence, and only tracking constant reference signals with zero steady-state error (and therefore with small steady-state error for slowly time-varying reference signals). This serves as motivation for the synthesis of other distributed algorithms. For clarity of presentation, we review continuous-time and discrete-time strategies separately.

Continuous-time dynamic average consensus algorithms

In the following, we introduce various continuous-time dynamic average consensus algorithms and discuss 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, however, as we will show similar to the algorithm (11) this requirement can be eliminated using alternative implementations.

Robustness to initialization

The requirement on precise initialization of the algorithm (11) makes it not robust to initialization errors. This is of particular relevance given that the initial value is transmitted over the communication channels, which might be subject to noise and other perturbations.

It is simple to see why the condition on the initial states is necessary. Consider applying the average signal $\mathbf{1}_N^\top \mathbf{U}(z)$ as the input in Figure 5. Since $\mathbf{1}_N^\top \mathbf{L} = 0$ for balanced graphs, the output of the Laplacian block is zero. Then $\mathbf{1}_N^\top \mathbf{X}(z) = \mathbf{1}_N^\top \mathbf{U}(z) - \mathbf{1}_N^\top \mathbf{p}_0$. For the average of the outputs to be the average of the reference signals, we need $\mathbf{1}_N^\top \mathbf{p}_0 = \mathbf{0}$. This can also be seen directly from the block diagram since the integrator states are directly connected to the output without first passing through the Laplacian (see [43]).

To eliminate the special initialization requirement and to induce robustness with respect to algorithm initialization, [44] proposes the following alternative dynamic average consensus algorithm

$$\dot{q}^i(t) = -\sum_{j=1}^{N} b_{ij}\,(x^i - x^j), \tag{17a}$$

$$\dot{x}^i = -\alpha\,(x^i - \mathsf{u}^i) - \sum_{j=1}^{N} a_{ij}\,(x^i - x^j) + \sum_{j=1}^{N} b_{ji}\,(q^i - q^j), \tag{17b}$$

$$q^i(0),\, x^i(0) \in \mathbb{R}, \qquad i \in \{1, \ldots, N\}, \tag{17c}$$

where $\alpha \in \mathbb{R}_{>0}$. Here, 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, L_{p}, labeled as *proportional* Laplacian and L_{I}, labeled as *integral* Laplacian. The compact representation of (17) is as follows

$$\dot{\mathbf{q}} = -\mathsf{L}_{\mathrm{I}}\,\mathbf{x}, \tag{18a}$$

$$\dot{\mathbf{x}} = -\alpha\,(\mathbf{x} - \mathbf{u}) - \mathsf{L}_{\mathrm{p}}\,\mathbf{x} + \mathsf{L}_{\mathrm{I}}^{\top}\mathbf{q}, \tag{18b}$$

which also reads as

$$\dot{\mathbf{x}} = -\alpha\,(\mathbf{x} - \mathbf{u}) - \mathsf{L}_{\mathrm{p}}\,\mathbf{x} - \mathsf{L}_{\mathrm{I}}^{\top}\int_0^t \mathsf{L}_{\mathrm{I}}\,\mathbf{x}(\tau)\,\mathrm{d}\tau.$$

The presence of the transposed integral Laplacian, $\mathsf{L}_{\mathrm{I}}^{\top}$, in (18b) requires each agent $i \in \{1, \ldots, N\}$ to know not only the entries in row i but also the column i of L_{I}, and receive information from the corresponding agents. Therefore, for directed graphs, (17) is not implementable (we come back to this point later). However, for undirected graph topologies this requirement is satisfied trivially as $\mathsf{L}_{\mathrm{I}}^{\top} = \mathsf{L}_{\mathrm{I}}$. The next result states the convergence properties of (17).

Theorem 0.3 (Convergence of (17) over strongly connected and weight-balanced digraphs for dynamic input signals [44])**.** *Let $\mathcal{G}$ be a strongly connected and weight-balanced digraph. Starting from any initial condition $q^i(0), x^i(0) \in \mathbb{R}$, for static inputs $\lim_{t\to\infty} \|\mathbf{e}(t)\| = 0$, while, for bounded inputs with bounded rates, $\|\mathbf{e}(t)\|$ is bounded.*

Using a time-domain analysis similar to that of algorithm (11), we can obtain further information on the ultimate tracking behavior of the algorithm (17). Consider the change of variables (13) and

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \mathbf{w}_{2:N} \end{bmatrix} = \mathbf{T}^{\top}\,\mathbf{q}. \tag{19}$$

This allows us to write (18) in the equivalent form

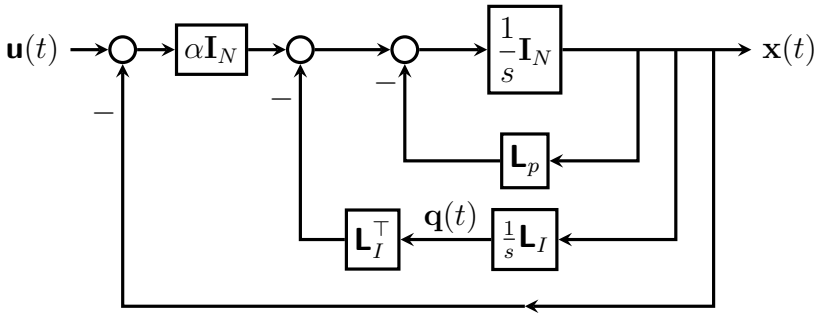
$$\dot{w}_1 = 0, \tag{20a}$$

$$\begin{bmatrix} \dot{\mathbf{w}}_{2:N} \\ \dot{\bar{e}}_1 \\ \dot{\bar{\mathbf{e}}}_{2:N} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{R}^{\top}\mathsf{L}_{\mathrm{I}}\mathbf{R} \\ 0 & -\alpha & 0 \\ \mathbf{R}^{\top}\mathsf{L}_{\mathrm{I}}^{\top}\mathbf{R} & \mathbf{0} & -\alpha\mathbf{I} - \mathbf{R}^{\top}\mathsf{L}_{\mathrm{p}}\mathbf{R} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \mathbf{w}_{2:N} \\ \bar{e}_1 \\ \bar{\mathbf{e}}_{2:N} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -\frac{1}{\sqrt{N}}\sum_{j=1}^{N} \mathsf{u}^j \\ \alpha\mathbf{R}^{\top}\mathbf{u} \end{bmatrix}. \tag{20b}$$

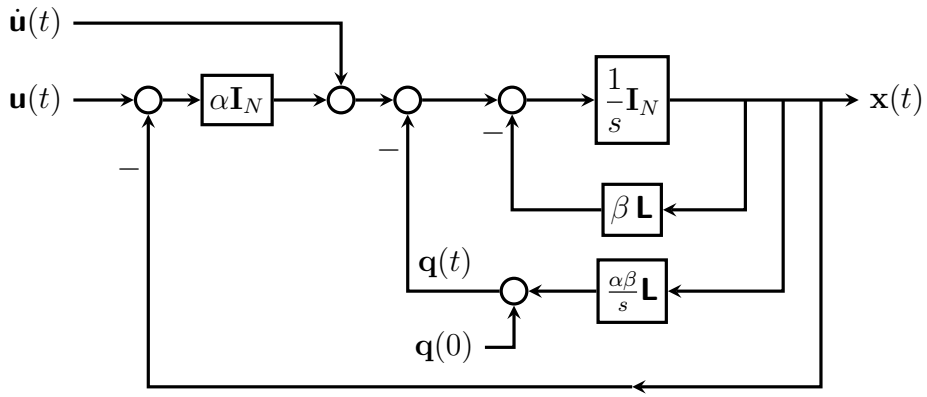
One can show that the matrix $\mathbf{A}$ in (20b) is Hurwitz. Therefore, using the ISS bound on the trajectories of LTI systems (see "Sidebar 3: Input-to-State Stability of LTI Systems"), the tracking error of each agent $i \in \{1, \dots, N\}$ while implementing (17) over a strongly connected and weight-balanced digraph is

$$|e^i(t)| \leq \kappa\, \mathrm{e}^{-\underline{\lambda} t} \left\| \begin{bmatrix} \mathbf{w}_{2:N}(0) \\ \bar{\mathbf{e}}(0) \end{bmatrix} \right\| + \frac{\kappa}{\underline{\lambda}} \sup_{0 \leq \tau \leq t} \sqrt{|\frac{1}{\sqrt{N}} \sum_{j=1}^{N} \mathsf{u}^j(\tau)|^2 + \alpha^2 \|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\mathbf{u}(\tau)\|^2}, \quad (21)$$

where $(\kappa, \underline{\lambda})$ can be computed from (S3) for matrix $\mathbf{A}$ in (20b). From this error bound, we observe that for bounded dynamic signals with bounded rate the algorithm (17) is guaranteed to track the dynamic average with an ultimately bounded error. Moreover, we can see that this algorithm does not need any special initialization.



(a) Continuous-time algorithm in (17)



(b) Continuous-time algorithm in (23)

Figure 6: Block diagram of the dynamic average consensus algorithms (18) and (23). Here, the reference signals are applied as inputs to the system.

Controlling the rate of convergence

A common feature of the dynamic average consensus algorithms presented above is that the rate of convergence is the same for all agents and is dictated by network topology, as well as some

algorithm parameters (see (14) and (21)). 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 pre-specify its desired worst rate of convergence, β, [45] proposes dynamic average consensus algorithms whose design incorporates two time scales. The 1*st-Order-Input Dynamic Consensus* (*FOI-DC*) algorithm is described as follows

$$\begin{cases} \epsilon\, \dot{q}^i = -\sum_{j=1}^N a_{ij}(z^i - z^j), \\ \epsilon\, \dot{z}^i = -(z^i + \beta\, \mathsf{u}^i + \dot{\mathsf{u}}^i) - \sum_{j=1}^N \mathsf{a}_{ij}(z^i - z^j) + \sum_{j=1}^N \mathsf{a}_{ji}(q^i - q^j), \end{cases} \tag{22a}$$

$$\dot{x}^i = -\beta\, x^i - z^i, \quad i \in \{1, \dots, N\} \tag{22b}$$

The fast dynamics here is (22a), and employs a small value for $\epsilon \in \mathbb{R}_{>0}$. The fast dynamics, which builds on the PI algorithm (17), is intended to generate the average of the sum of the dynamic input and its first derivative. The slow dynamics (22b) then uses the signal generated by the fast dynamics to track the average of the reference signal across the network at a pre-specified smaller rate $\beta \in \mathbb{R}_{>0}$. The novelty here 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 towards the input average.

The technical approach used in [45] to study the convergence of (22) is based on singular perturbation theory [46, Chapter 11]. Similar to the dynamic average consensus algorithm (17), (22) does not require any specific initialization. The following result gives the convergence guarantees of *FOI-DC*.

Theorem 0.4 (Convergence of *FOI-DC* [45])**.** *Let $\mathcal{G}$ be a strongly connected and weight-balanced digraph. Assume that in FOI-DC algorithm the first and the second derivatives of the input signal u^i at each agent $i \in \{1, \dots, N\}$ are continuous and bounded for $t \in \mathbb{R}_{\geq 0}$. Then, there exists $\epsilon^\star \in \mathbb{R}_{>0}$ such that, for all $\epsilon \in (0, \epsilon^\star]$, starting from any initial conditions $\mathbf{x}(0), \mathbf{z}(0), \boldsymbol{\nu}(0) \in \mathbb{R}^N$, the state x^i, $i \in \{1, \dots, N\}$, of algorithm* (22) *converges exponentially fast with rate β to an $O(\epsilon)$-neighborhood of $\mathsf{u}^{avg}(t)$,*

$$\lim_{t\to\infty} \left|x^i(t) - \mathsf{u}^{avg}(t)\right| \leq \left|x^i(0) - \mathsf{u}^{avg}(t)\right|\, e^{-\beta t} + O(\epsilon), \quad i \in \{1, \dots, N\}.$$

Using time-domain analysis, we can make more precise the information about the ultimate tracking behavior of algorithm(17). For convenience, we apply the change of variables (13), (19) along with $\mathbf{e}_z = \mathbf{T}^\top(\mathbf{z} + \beta \frac{1}{N}\sum_{j=1}^N \mathsf{u}^j \mathbf{1}_N + \frac{1}{N}\sum_{j=1}^N \dot{\mathsf{u}}^j \mathbf{1}_N)$ to write the *FOI-DC* algorithm as follows

$$\begin{aligned} \dot{w}_1 &= 0, \\ \begin{bmatrix} \dot{\mathbf{w}}_{2:N} \\ \dot{\mathbf{e}}_z \end{bmatrix} &= \epsilon^{-1} \underbrace{\begin{bmatrix} \mathbf{0} & \begin{bmatrix} \mathbf{0} & -\mathbf{R}^\top \mathbf{L}_{\mathrm{I}} \mathbf{R} \end{bmatrix} \\ \begin{bmatrix} 0 \\ \mathbf{R}^\top \mathbf{L}_{\mathrm{I}}^\top \mathbf{R} \end{bmatrix} & \begin{bmatrix} -1 & 0 \\ \mathbf{0} & -\mathbf{I} - \mathbf{R}^\top \mathbf{L}_{\mathrm{p}} \mathbf{R} \end{bmatrix} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \mathbf{w}_{2:N} \\ \mathbf{e}_z \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{T}^\top \mathbf{f} \end{bmatrix}, \\ \dot{\mathbf{e}} &= -\beta\, \mathbf{e} - \mathbf{e}^z, \end{aligned}$$

where $\mathbf{f}(t) = -\epsilon^{-1}(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)(\beta\,\mathbf{u} + \dot{\mathbf{u}}) + \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top(\beta\,\dot{\mathbf{u}} + \ddot{\mathbf{u}})$. Using the ISS bound on the trajectories of LTI systems, see "Sidebar 3: Input-to-State Stability of LTI Systems", the tracking error of each agent $i \in \{1, \dots, N\}$ while implementing *FOI-DC* algorithm with an $\epsilon \in \mathbb{R}_{>0}$ is as follows, $i \in \{1, \dots, N\}$,

$$|e_x^i(t)| \leq \mathrm{e}^{-\beta\,t}|e_x^i(0)| + \frac{\kappa}{\beta}\sup_{0\leq\tau\leq t}\Big(\mathrm{e}^{-\epsilon^{-1}\underline{\lambda}\,t}\Big\|\begin{bmatrix}\mathbf{w}_{2:N}(0)\\ \mathbf{e}_z(0)\end{bmatrix}\Big\| + \frac{\epsilon}{\underline{\lambda}}\sup_{0\leq\tau\leq t}\|\mathbf{f}\|\Big),$$

From this error bound, we observe that for bounded dynamic signals with bounded first and second derivative *FOI-DC* algorithm is guaranteed to track the dynamic average with an ultimately bounded error. Also, using small $\epsilon \in \mathbb{R}_{>0}$ results in dynamics (23a) to have a higher decay rate and therefore, the dominant rate of convergence of *FOI-DC* algorithm to be determined by β.

Implementation on Directed Graphs with a Tunable Error Bound

As we have observed above, the algorithm (17) is not implementable over directed graphs, since it requires information exchange with both in- and out-neighbors, and these sets are generally different. Here, we introduce a design proposed in [17] that address this problem. The algorithm also uses proportional and integral agreement feedback and its implementation does not require the agents to know their respective columns of the Laplacian. For $i \in \{1, \dots, N\}$, consider

$$\dot{q}^i = \alpha\beta\sum\nolimits_{j=1}^{N}\mathsf{a}_{ij}(x^i - x^j), \tag{23a}$$

$$\dot{x}^i = \dot{\mathsf{u}}^i - \alpha(x^i - \mathsf{u}^i) - \beta\sum\nolimits_{j=1}^{N}\mathsf{a}_{ij}(x^i - x^j) - q^i, \tag{23b}$$

$$x^i(0), q^i(0) \in \mathbb{R} \;\; s.t. \;\sum\nolimits_{j=1}^{N} q^j(0) = 0, \tag{23c}$$

In compact form, this algorithm can be expressed as follows

$$\begin{aligned}\dot{\mathbf{q}} &= \alpha\,\beta\,\mathsf{L}\,\mathbf{x},\\ \dot{\mathbf{x}} &= \dot{\mathbf{u}} - \alpha\,(\mathbf{x} - \mathbf{u}) - \beta\,\mathsf{L}\,\mathbf{x} - \mathbf{q},\end{aligned}$$

which can be equivalently written as

$$\dot{\mathbf{x}} = \dot{\mathbf{u}} - \alpha\,(\mathbf{x} - \mathbf{u}) - \beta\,\mathsf{L}\,\mathbf{x} - \alpha\,\beta\int_0^t \mathsf{L}\,\mathbf{x}(\tau)\,\mathrm{d}\tau.$$

As we did for algorithm (11), we can use a change of variables $p^i = \mathsf{u}^i - x^i$, to write this algorithm in a form whose implementation does not require the knowledge of the derivative of the reference

signals as follows,

$$\begin{aligned}
\dot{q}^i &= \alpha\beta \sum\nolimits_{j=1}^{N} \mathsf{a}_{ij}(x^i - x^j),\\
\dot{p}^i &= p^i - \beta \sum\nolimits_{j=1}^{N} \mathsf{a}_{ij}(x^i - x^j) - q^i,\\
x^i &= u^i - p^i,\\
&\quad p^i(0), q^i(0) \in \mathbb{R} \;\; s.t. \;\; \sum\nolimits_{j=1}^{N} q^j(0) = 0, \quad i \in \{1, \dots, N\}.
\end{aligned}$$

Note that the initialization condition $\sum_{i=1}^{N} q^i(0) = 0$ can be easily satisfied if each agent $i \in \{1, \dots, N\}$ starts at $q^i(0) = 0$. This is a mild requirement because q^i is an internal state for agent i and therefore is not affected by communication errors. This initialization condition however, limits the use of algorithm (23) in applications where agents join the network at different points in time. The next result states the convergence properties of (23). We refer the reader to [17] for the proof of this statement which is established using the time domain analysis we implemented to analyze the algorithm we reviewed so far.

Theorem 0.5 (Convergence of (23) over strongly connected and weight-balanced digraphs for dynamic input signals [17])**.** *Let $\mathcal{G}$ be a strongly connected and weight-balanced digraph. Let the agent inputs satisfy $\|(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\dot{\mathbf{u}}\|_{ess} = \gamma < \infty$. Then, for any $\alpha, \beta \in \mathbb{R}_{>0}$, the trajectories of algorithm* (23) *satisfy*

$$\lim_{t\to\infty} \left| x^i(t) - \mathsf{u}^{avg}(t) \right| \leq \frac{\gamma}{\beta\hat{\lambda}_2}, \qquad i \in \{1, \dots, N\}. \tag{24}$$

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 (24) indicates that we can use the parameter β to control the tracking error size.

Discrete-time dynamic average consensus algorithms

In the following, we introduce various discrete-time dynamic consensus algorithms, provide historical context on their design, and discuss to what extent they address the shortcomings identified for the initial design (16). Table 2 summarizes the arguments of the driving command of these algorithms in (2) and their special initialization requirements.

Robustness to initialization

The dynamic average consensus algorithm (16) is not robust to initial conditions, meaning that the final consensus value depends on the initial states of the agents. In particular, the initial states

Table 2: Arguments of the driving command in (2) for the reviewed discrete-time dynamic average consensus algorithms together with their initialization requirements. Note that some algorithms use the future input u_{k+1} in the update, but can be rewritten to only require the current input u_k.

Algorithm	(15)	(25)	(30)	(31)	(32)
$J^i(t)$	$\{x_k^i, \mathsf{u}_k^i, \mathsf{u}_{k+1}^i\}$	$\{x_k^i, \mathsf{u}_k^i, \mathsf{u}_{k+1}^i\}$	$\{p_k^i, \mathsf{u}_k^i\}$	$\{p_k^i, \mathsf{u}_k^i\}$	$\{p_k^i, q_k^i, \mathsf{u}_k^i\}$
$\{I^j(t)\}_{j\in\mathcal{N}_{\text{out}}^i}$	$\{x_k^j\}_{j\in\mathcal{N}_{\text{out}}^i}$	$\{x_k^j\}_{j\in\mathcal{N}_{\text{out}}^i}$	$\{p_k^j\}_{j\in\mathcal{N}_{\text{out}}^i}$	$\{p_k^j\}_{j\in\mathcal{N}_{\text{out}}^i}$	$\{x_k^i, p_k^j\}_{j\in\mathcal{N}_{\text{out}}^i}$
Initialization Requirement	$x_0^i = \mathsf{u}_0^i$	none	none	$\sum_{j=1}^N p_0^j$ close to zero	none

must be initialized such that $\mathbf{1}_N^\top \mathbf{x}_0 = \mathbf{1}_N^\top \mathbf{u}_0$ (or, equivalently, $\mathbf{1}_N^\top \mathbf{p}_0 = 0$) in order for the output to converge to the correct average. Correct initialization not only requires synchronization of the algorithm start time, but also means that a single communication fault can introduce errors which propagate through the system and affect the steady-state consensus value. Therefore, it is desired to have a dynamic average consensus algorithm which is *robust to initial conditions*, meaning that the steady-state error is independent of the initial states. Next, we discuss several methods of modifying the algorithm to obtain robustness to initial conditions, although each method has drawbacks as well. Note that robustness to initial conditions can be verified using the necessary and sufficient conditions in [47].

Move the pole away from $z = 1$

One method of modifying the dynamic average consensus algorithm in Figure 5 to make it robust to initial conditions is to move the open-loop pole from $z = 1$ to $z = \gamma$ where $|\gamma| < 1$. The resulting estimator, whose block diagram is shown in Figure 7, is given by

$$x_{k+1}^i = \gamma x_k^i - k_I \sum_{j=1}^N a_{ij}(x_k^i - x_k^j) + (\mathsf{u}_{k+1}^i - \gamma\, \mathsf{u}_k^i), \quad x_0^i \in \mathbb{R}, \quad i \in \{1, \dots, N\}, \tag{25}$$

or equivalently,

$$p_{k+1}^i = \gamma\, p_k^i + k_I \sum_{j=1}^N a_{ij}(x_k^i - x_k^j), \qquad p_0^i \in \mathbb{R}, \qquad i \in \{1, \dots, N\} \tag{26a}$$

$$x_k^i = u_k^i - p_k^i. \tag{26b}$$

A bound on the error of the trajectories is given in Theorem 0.6.

Theorem 0.6 (Proportional estimator)**.** *Suppose that the communication graph is constant, strongly-connected, and weight-balanced. Let $\mathbf{R} \in \mathbb{R}^{N\times(N-1)}$ be such that $\left[\frac{1}{\sqrt{N}}\mathbf{1}_N \quad \mathbf{R}\right]$ is orthonormal. Suppose there exists $\mathbf{P} \in \mathbb{R}^{N\times N}$ and $\rho \in \mathbb{R}$ such that*

$$(\gamma \mathbf{I}_N - k_I \mathbf{R}^\top \mathbf{L}\mathbf{R})^\top \mathbf{P} (\gamma \mathbf{I}_N - k_I \mathbf{R}^\top \mathbf{L}\mathbf{R}) - \rho^2\, \mathbf{P} \leq 0, \quad \mathbf{P} > 0, \quad \rho \geq 0. \tag{27}$$

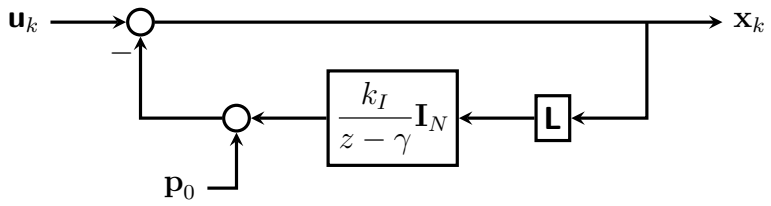


Figure 7: Block diagram of the dynamic average consensus algorithm with the pole at $z = \gamma$. When $\gamma = 1$, the error converges to zero (provided that the internal state is initialized such that $\mathbf{1}_N^\top \mathbf{p}_0 = 0$). On the other hand, the error converges to a finite value when $|\gamma| < 1$ independent of the initial state. In particular, no value of γ provides both zero steady-state error for constant input signals and robustness to initial conditions.

Then the error of the dynamic average consensus algorithm (25) *satisfies the bound*

$$\begin{aligned}\|\mathbf{e}_k\|^2 \leq &\left(\frac{\gamma^k}{\sqrt{N}}\,\|\mathbf{1}_N^\top(\mathbf{u}_0 - \mathbf{x}_0)\|\right)^2 \\ &+ \left(\sqrt{\text{cond}(\mathbf{P})}\,\rho^k\,\|(\mathbf{I}_N - \tfrac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)(\mathbf{u}_0 - \mathbf{x}_0)\|\right)^2 \\ &+ \left(\frac{1-\rho^k}{1-\rho}\sup_{0\leq m<k}\|(\mathbf{I}_N - \tfrac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)(\mathbf{u}_{k+1} - \gamma\,\mathbf{u}_k\|\right)^2 \end{aligned} \tag{28}$$

for all $k \in \mathbb{Z}_{\geq 0}$.

Proof. The dynamic average consensus algorithm (25) can be written compactly as

$$\mathbf{x}_{k+1} = (\gamma\mathbf{I}_N - k_I\mathbf{L})\mathbf{x}_k + (\mathbf{u}_{k+1} - \gamma\,\mathbf{u}_k).$$

Defining the error as $\mathbf{e}_k := \mathbf{x}_k - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top\mathbf{u}_k$, the error dynamics are

$$\mathbf{e}_{k+1} = (\gamma\mathbf{I}_N - k_I\mathbf{L})\mathbf{e}_k + (\mathbf{I}_N - \tfrac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)(\mathbf{u}_{k+1} - \gamma\,\mathbf{u}_k).$$

Define the transformation matrix $\mathbf{T} = \begin{bmatrix}\frac{1}{\sqrt{N}}\mathbf{1}_N & \mathbf{R}\end{bmatrix}$ where $\mathbf{R} \in \mathbb{R}^{N\times(N-1)}$ is such that $\mathbf{T}$ is orthonormal, and consider the change of variables

$$\bar{\mathbf{e}}_k = \begin{bmatrix}\bar{e}_k^1 \\ \bar{\mathbf{e}}_k^{2:N}\end{bmatrix} = \mathbf{T}^\top\mathbf{e}_k. \tag{29}$$

In the new coordinates, (25) takes the form

$$\begin{aligned}\bar{e}_{k+1}^1 &= \gamma\,\bar{e}_k^1, & \bar{e}_0^1 &= \frac{1}{\sqrt{N}}\mathbf{1}_N^\top(\mathbf{x}_0 - \mathbf{u}_0), \\ \bar{\mathbf{e}}_{k+1}^{2:N} &= (\gamma\mathbf{I}_N - k_I\mathbf{R}^\top\mathbf{L}\mathbf{R})\bar{\mathbf{e}}_{2:N} + \mathbf{R}^\top(\mathbf{u}_{k+1} - \gamma\,\mathbf{u}_k), & \bar{\mathbf{e}}_0^{2:N} &= \mathbf{R}^\top\mathbf{x}_0.\end{aligned}$$

The norm of the error is then

$$\|\mathbf{e}_k\|^2 = \|\bar{\mathbf{e}}_k\|^2 = |\bar{e}_k^1|^2 + \|\bar{\mathbf{e}}_k^{2:N}\|^2,$$

and the bound (28) follows from applying the ISS bound for the trajectories of discrete-time systems in "Sidebar 3: Input-to-State Stability of LTI Systems" to each of the two error terms. □

The bound (28) provides several insights into the dynamic average consensus algorithm (25). First, the effects of the initial conditions decay exponentially to zero with rate γ in the consensus direction (as long as $|\gamma| < 1$) and with rate ρ in the disagreement directions (as long as $|\rho| < 1$). Second, the algorithm achieves zero steady-state error for constant reference signals only when $\gamma = 1$, that is, when there is an open-loop pole at $z = 1$. By moving the pole, the steady-state error is no longer zero.

A slight modification of this approach, however, can achieve both robustness to initial conditions and zero steady-state error [48]. The idea is to use time-varying dynamics to have the pole drift towards $z = 1$. In particular, the pole at time k is at $z = \gamma_k$ where $\gamma_k \in (0, 1)$ and $\gamma_k \to 1$ as $k \to \infty$. This approach is attractive in theory since it is robust to initial conditions and achieves zero steady-state error, but does not work well in practice. If the dynamic average consensus algorithm is reinitialized (for example, if the graph changes) when the pole is still far from $z = 1$, then the output converges quickly. As the pole approaches $z = 1$, however, the dynamic average consensus algorithm takes longer and longer to recover from initialization errors. As the pole drifts towards $z = 1$, robustness to initial conditions slowly deteriorates while the steady-state error is improved, but at any finite time there is a trade-off between the two properties.

Swap the order of the integrator and the Laplacian matrix

The dynamic average consensus algorithm can also be made robust to initial conditions with a simple modification of the block diagram. Consider swapping the order of the integrator and the Laplacian block as shown in Figure 8. The integrator states now pass through the Laplacian before reaching the output and therefore have no affect in the consensus direction. The dynamic average consensus algorithm is then

$$p_{k+1}^i = p_k^i + x_k^i, \qquad p_0^i \in \mathbb{R}, \qquad i \in \{1, \dots, N\} \tag{30a}$$

$$x_k^i = u_k^i - k_I \sum_{j=1}^{N} a_{ij}(p_k^i - p_k^j), \tag{30b}$$

and the error converges exponentially to zero independent of the initial states $\mathbf{p}_0$.

By making the dynamic average consensus algorithm robust to initial conditions, however, another problem has been introduced; the dynamic average consensus algorithm is no longer internally stable. To see why this happens, consider the dynamic average consensus algorithm in steady-state. The output of the dynamic average consensus algorithm has then converged to the average of the reference signals, and this constant (nonzero) value is injected directly into the integrator. The integrator states grow unbounded as ramp signals. This is undesirable when the dynamic average consensus algorithm is used to track the reference signals over a long period of time as the internal states will eventually cause overflow.

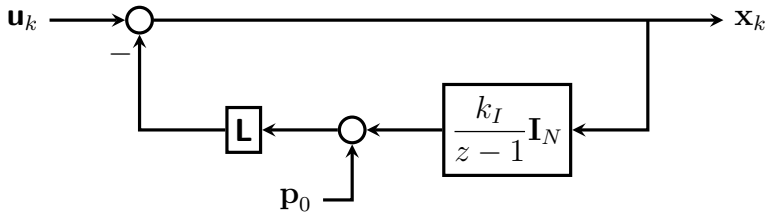


Figure 8: Block diagram of the dynamic average consensus algorithm with the integrator and Laplacian block swapped. This version is robust to initial conditions since the initial condition is not directly connected to the output in the consensus direction. For constant reference signals, however, the algorithm is not internally stable since the integrator state grows like a ramp as $k \to \infty$.

Use the nonlinear Laplacian operator

Yet another approach for obtaining robustness to initial conditions is to change the state space of the dynamic average consensus algorithm using the nonlinear Laplacian operator [49]. Recall that the problem with the dynamic dynamic average consensus algorithm in Figure 8 is that the integrator states grow unbounded when the outputs have converged to the average of the reference signals. To fix this, the state space on each agent can be changed from the real line to the unit circle. Doing so introduces nonlinearities into the dynamics, and the integrator states now take values on the compact manifold $\mathbb{T}$ instead of the real line, where $\mathbb{T}$ denotes the unit circle. The dynamics must be modified to make this work, but the result is that the integrator states are automatically bounded regardless of the other signals in the system.

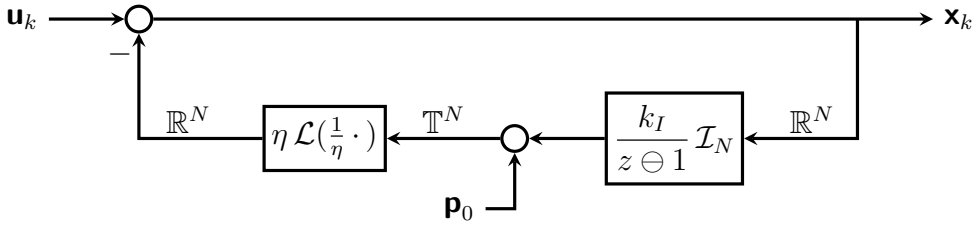


Figure 9: Block diagram of the nonlinear dynamic average consensus algorithm where $\mathcal{L}(x) = \mathbf{B}Wf(\mathbf{B}^\top\mathbf{x})$ is the nonlinear Laplacian operator and $\mathcal{I} : \mathbb{R} \to \mathbb{T}$ is the covering projection $s \mapsto \mathcal{I}(s) = \exp\{s\sqrt{-1}\}$ for $s \in \mathbb{R}$. There exists an open set of initial conditions from which the error converges to zero, but the set is not the entire space. In other words, the dynamic average consensus algorithm is *locally* stable but not *globally* stable.

Similar to the Laplacian matrix, the Laplacian operator $\mathcal{L} : \mathbb{T}^n \to \mathbb{R}^n$ associated with an undirected graph is defined as

$$\mathcal{L}(\mathbf{x}) = \mathbf{B}Wf(\mathbf{B}^\top\mathbf{x})$$

where $f : \mathbb{T} \to \mathbb{R}$ is odd with $f(0) = 0$, $\mathbf{B}$ is the oriented incidence matrix of the graph, and W is a diagonal matrix of edge weights. Here, f is interpreted as acting element-wise on vector

arguments to produce vector values, and $B^\top$ is interpreted as a $\mathbb{Z}$-linear map from $\mathbb{T}^N$ to $\mathbb{T}^m$. Note that the Laplacian matrix is given by $\mathbf{L} = \mathbf{B}W\mathbf{B}^\top$. Let $\mathcal{I} : \mathbb{R} \to \mathbb{T}$ denote the covering projection $s \mapsto \mathcal{I}(s) = \exp\{s\sqrt{-1}\}$ for $s \in \mathbb{R}$. Then the nonlinear dynamic average consensus algorithm is given by the block diagram in Figure 9, and the time-domain equations are

$$p_{k+1}^i = p_k^i \oplus \mathcal{I}(k_I\, x_k), \qquad p_0^i \in \mathbb{T}, \qquad i \in \{1, \dots, N\} \tag{31a}$$

$$x_k^i = u_k^i - \eta \sum_{j=1}^{N} a_{ij}\, f\big(\tfrac{1}{\eta}(p_k^i - p_k^j)\big). \tag{31b}$$

Although the nonlinear dynamic average consensus algorithm is internally stable, the nonlinearities make it no longer globally exponentially stable. The dynamic average consensus algorithm is locally robust to initial conditions (that is, there is an open set of initial conditions from which the dynamic average consensus algorithm converges with zero steady-state error), but is not globally robust to initial conditions (the set of initial conditions for which the system converges is not the entire space).

Theorem 0.7 (Nonlinear estimator [49])**.** *Suppose the following:*

- *the communication graph is constant, connected, and undirected,*
- *the reference signals are constant,*
- *the phase coupling function $f : \mathbb{T} \to \mathbb{R}$ is odd with $f(0) = 0$, and there exists $b \in (0, \pi)$ such that $f(\theta) = \theta$ when $|\theta| \leq b$, and*
- *η is sufficiently large, and $k_I > 0$.*

Then there exists an open set of initial conditions p_0^i around the origin from which the agreement states x_k^i of the nonlinear dynamic average consensus algorithm (31) *converges exponentially to the average $\mathbf{u}^{avg}$.*

Introduce a second Laplacian block

The final approach considered achieves both robustness to initial conditions and internal stability using linear time-invariant dynamics. To do this, a second Laplacian block is introduced in the block diagram as shown in Figure 10. The integrator states pass through a Laplacian block before reaching the output so the dynamic average consensus algorithm is robust to initial conditions, and the output passes through a Laplacian block before reaching the integrator so the dynamic average consensus algorithm is internally stable. The feedback loop contains both proportional and integral terms, so this is referred to as the *proportional-integral (PI) dynamic average consensus*

algorithm [44]. For reference, the time-domain equations describing the PI estimator are

$$q_{k+1}^i = \gamma\, q_k^i + k_p \sum_{j=1}^{N} a_{ij}\big((x_k^i - x_k^j) + (p_k^i - p_k^j)\big), \tag{32a}$$

$$p_{k+1}^i = p_k^i + k_I \sum_{j=1}^{N} a_{ij}(x_k^i - x_k^j), \tag{32b}$$

$$x_k^i = u_k^i - q_k^i, \qquad p_0^i, q_0^i \in \mathbb{R}, \qquad i \in \{1, \dots, N\}. \tag{32c}$$

The convergence properties of the PI estimator are further discussed in the following section.

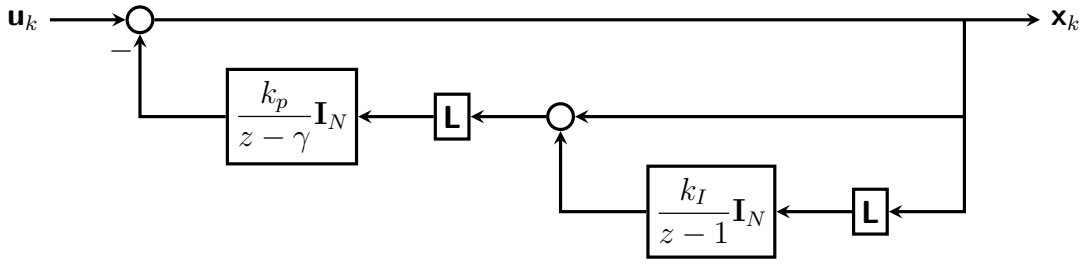


Figure 10: Block diagram of the proportional-integral dynamic average consensus algorithm (32) with parameters $\gamma, k_p, k_I \in \mathbb{R}$. The integrator states pass through the graph Laplacian before reaching the output, so the dynamic average consensus algorithm is robust to initial conditions. Also, the output passes through the Laplacian before reaching the integrator, so the dynamic average consensus algorithm is internally stable.

Accelerating the convergence rate

The error of each discrete-time dynamic average consensus algorithm presented above converges linearly to zero (sometimes requiring specific initialization). The rate of convergence depends on the structure of the underlying communication graph as well as the algorithm parameters. How to design dynamic average consensus algorithms to accelerate the convergence has received much attention in the literature [43, 50, 51, 52, 53, 54, 55, 56, 57, 58]. Here, we give a simple method for choosing the parameters in order to optimize the convergence rate. We also show how to further accelerate the convergence by introducing extra dynamics into the dynamic average consensus algorithm.

For simplicity of exposition, we assume that the communication graph is constant, connected, and undirected. The Laplacian matrix is then symmetric and therefore has real eigenvalues. Since the graph is connected, the smallest eigenvalue is $\lambda_1 = 0$ and all the other eigenvalues are strictly positive, in other words, $\lambda_2 > 0$. Furthermore, we 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).

First, consider the dynamic average consensus algorithm from Figure 5b. This algorithm has a

single parameter k_I. 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 5b 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

$$0 = 1 + \lambda \frac{k_I}{z - 1}. \tag{33}$$

To observe how the pole moves as a function of the Laplacian eigenvalue, root locus techniques from LTI systems theory can be used. Figure 12a shows the root locus of (33) as a function of λ. The dynamic average consensus algorithm poles are then the points on the root locus at gains λ_i for $i \in \{1, \dots, N\}$ where λ_i are the eigenvalues of the graph Laplacian. To optimize the convergence rate, the system should be 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. Since the pole starts at $z = 1$ and moves left as λ increases, the convergence rate is 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

$$k_I = \frac{2}{\lambda_2 + \lambda_N} \quad \text{and} \quad \rho = \frac{\lambda_N - \lambda_2}{\lambda_N + \lambda_2}. \tag{34}$$

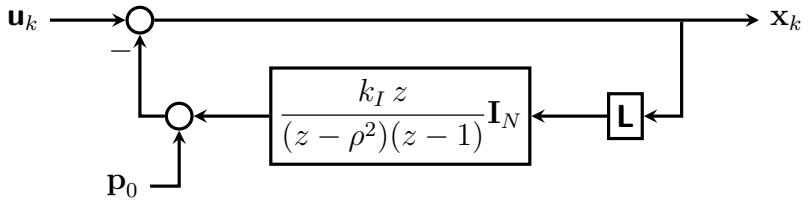
While the previous choice of parameters optimizes the convergence rate, even faster convergence can be achieved by introducing extra dynamics into the dynamic average consensus algorithm. Consider the accelerated dynamic average consensus algorithm in Figure 11a, given by

$$p_{k+1}^i = (1 + \rho^2) p_k^i - \rho^2 p_{k-1}^i + k_I \sum_{j=1}^{N} a_{ij}(x_k^i - x_k^j), \qquad p_0^i \in \mathbb{R}, \qquad i \in \{1, \dots, N\}, \tag{35a}$$

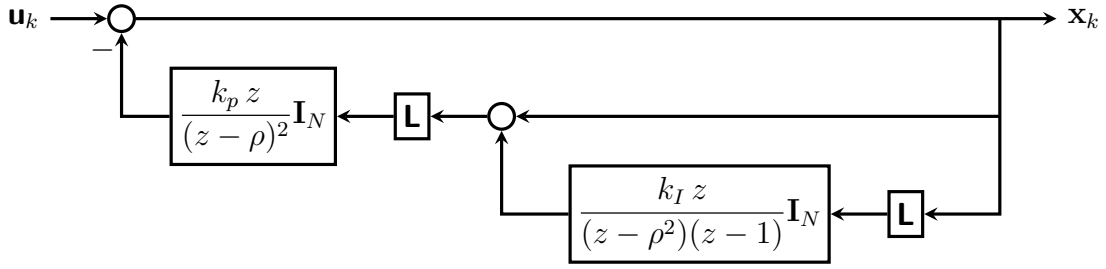
$$x_k^i = u_k^i - p_k^i. \tag{35b}$$

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 keep track of 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 below.

Once again, the root locus can be used to design the parameters to optimize the convergence rate. Figure 12b shows the root locus of the accelerated dynamic average consensus algorithm (35).



(a) Accelerated dynamic average consensus algorithm



(b) Accelerated proportional-integral dynamic average consensus algorithm

Figure 11: Block diagram of accelerated dynamic average consensus algorithms. Extra dynamics are used to accelerate the convergence rate. When the graph is connected and balanced, and upper and lower bounds on the eigenvalues of the graph Laplacian are known, closed-form solutions for the parameters which optimize the convergence rate are known.

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

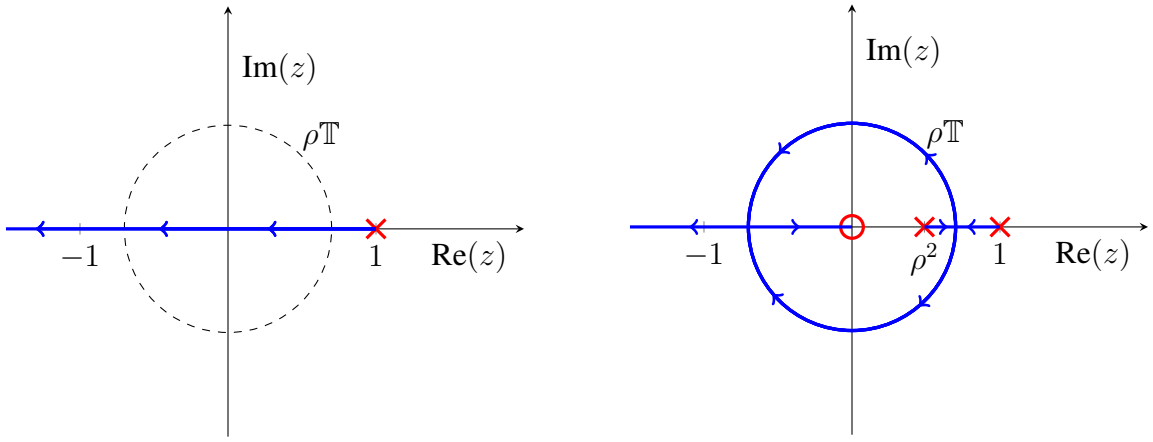
$$k_I = \frac{4}{(\sqrt{\lambda_2} + \sqrt{\lambda_N})^2}$$

and the corresponding convergence rate

$$\rho = \frac{\sqrt{\lambda_N} - \sqrt{\lambda_2}}{\sqrt{\lambda_N} + \sqrt{\lambda_2}}. \tag{36}$$

The convergence rate of both the standard (Eq. (16)) and accelerated (Eq. (35)) versions of the dynamic average consensus algorithm are plotted in Figure 13a as a function of the ratio λ_2/λ_N.

Root locus techniques can also be used to optimize the convergence rate of other dynamic average consensus algorithms. Since the dynamic average consensus algorithms (16) and (35) have only one Laplacian block in the block diagram, the resulting root loci are linear in the Laplacian eigenvalues. For the proportional-integral 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 [43], even



(a) Standard version using $\frac{k_I}{z-1}$.

(b) Accelerated version using $\frac{k_I z}{(z-\rho^2)(z-1)}$.

Figure 12: Root locus design of the integral dynamic average consensus algorithm. 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. Then the dynamic average consensus algorithm converges linearly with rate ρ.

for the accelerated version using extra dynamics, given by

$$q_{k+1}^i = 2\rho\, q_k^i - \rho^2 q_{k-1}^i + k_p \sum_{j=1}^N a_{ij}\big((x_k^i - x_k^j) + (p_k^i - p_k^j)\big), \tag{37a}$$

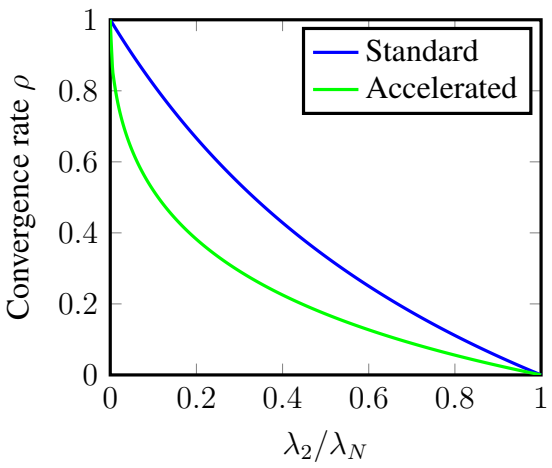
$$p_{k+1}^i = (1+\rho^2)p_k^i - \rho^2 p_{k-1}^i + k_I \sum_{j=1}^N a_{ij}(x_k^i - x_k^j), \tag{37b}$$

$$x_k^i = u_k^i - q_k^i, \qquad p_0^i, q_0^i \in \mathbb{R}, \qquad i \in \{1, \dots, N\}, \tag{37c}$$

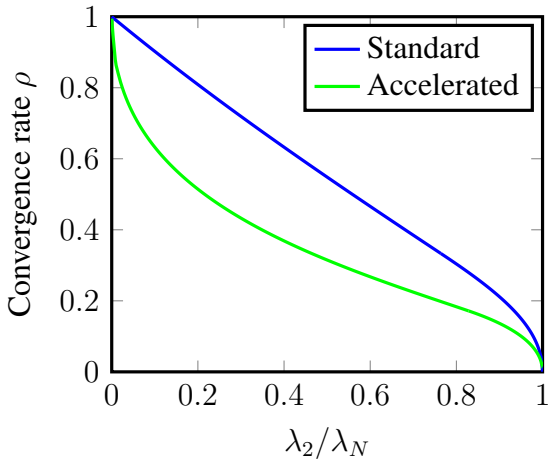
whose block diagram is in Figure 11b. The resulting convergence rate is plotted in Figure 13b. The convergence rate of the proportional-integral dynamic average consensus algorithm is slower than that of the non-robust algorithm, thought the proportional-integral algorithm is robust to initial conditions.

The following result summarizes the parameter choices which optimize the convergence rate for several DT dynamic average consensus algorithms. The results for the first two algorithms follow from the previous discussion, while details of the results for the last two algorithms can be found in [43].

Theorem 0.8 (Optimal convergence rates of DT dynamic average consensus algorithms)**.** *Suppose that the communication graph is constant, connected, and undirected, and that the reference signal* $\mathbf{u}^i$ *at each agent* $i \in \{1, \dots, N\}$ *is a constant scalar. Define the ratio* $\lambda_r := \lambda_2/\lambda_N$.



(a) Convergence rate of standard (Eq. (16)) and accelerated (Eq. (35)) dynamic average consensus algorithms which are *not* robust to initial conditions.



(b) Convergence rate of standard (Eq. (32)) and accelerated (Eq. (37)) proportional-integral dynamic average consensus algorithms which are robust to initial conditions.

Figure 13: Convergence rate ρ as a function of λ_2/λ_N for standard (blue) and accelerated (green) dynamic average consensus algorithms. The accelerated dynamic average consensus algorithms in Figure 11 use extra dynamics to enhance the convergence rate. The graph is assumed to be constant, connected, and undirected with Laplacian eigenvalues λ_i for $i \in \{1, \dots, N\}$. In each case, the convergence rate of the proportional-integral dynamic average consensus algorithm is slower than that of the non-robust dynamic average consensus algorithm, although it is also robust to initial conditions. Closed-form expressions for the rates and algorithm parameters are given in Theorem 0.8.

Then the agreement states x_k^i, $i \in \{1, \dots, N\}$ of the following dynamic average consensus algorithms converge to $\mathbf{u}^{\text{avg}}$ exponentially with rate ρ if the average of the initial integrator states is zero (that is, $\sum_{i=1}^N p_0^i = 0$) and the following parameters are used:

Dynamic average consensus algorithm (16)

$$\rho = \frac{\lambda_N - \lambda_2}{\lambda_N + \lambda_2}, \qquad k_I = \frac{2}{\lambda_2 + \lambda_N}$$

Accelerated dynamic average consensus algorithm (35)

$$\rho = \frac{\sqrt{\lambda_N} - \sqrt{\lambda_2}}{\sqrt{\lambda_N} + \sqrt{\lambda_2}}, \qquad k_I = \frac{4}{(\sqrt{\lambda_2} + \sqrt{\lambda_N})^2}$$

Furthermore, the agreement states x_k^i, $i \in \{1, \dots, N\}$ of the following dynamic average consensus algorithms converge to $\mathbf{u}^{\text{avg}}$ exponentially with rate ρ if the following parameters are used (regardless of the initial states):

Proportional-integral dynamic average consensus algorithm (32)

$$\rho = \begin{cases} \rho_1, & 0 < \lambda_r \leq 3 - \sqrt{5} \\ \rho_2, & 3 - \sqrt{5} < \lambda_r \leq 1 \end{cases} \qquad k_I = \frac{1-\rho}{\lambda_2} \qquad k_p = \frac{1}{\lambda_N} \frac{\rho(1-\rho)\lambda_r}{\rho + \lambda_r - 1}$$

where

$$\rho_1 = \frac{8 - 8\lambda_r + \lambda_r^2}{8 - \lambda_r^2}, \qquad \rho_2 = \frac{\sqrt{(1-\lambda_r)(4 + \lambda_r^2(5-\lambda_r))} - \lambda_r(1-\lambda_r)}{2(1+\lambda_r^2)}$$

Accelerated proportional-integral dynamic average consensus algorithm (37)

$$\rho = \begin{cases} \rho_1, & 0 < \lambda_r \leq 2(\sqrt{2}-1) \\ \rho_2, & 2(\sqrt{2}-1) < \lambda_r \leq 1 \end{cases} \qquad k_I = \frac{(1-\rho)^2}{\lambda_2} \qquad k_p = \beta k_I$$

where

$$\rho_1 = \frac{4 - \beta + 4(1 - \sqrt{4-\beta})}{\beta}, \qquad \rho_2 = \frac{1 - \beta - 2(1 - \sqrt{\beta})}{1 - \beta}$$

and $\beta = 2 + 2\sqrt{1-\lambda_r} - \lambda_r$

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. In this section, we describe 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, we consider reference signals which either (1) have a known model, (2) are bandlimited, or (3) have bounded derivatives.

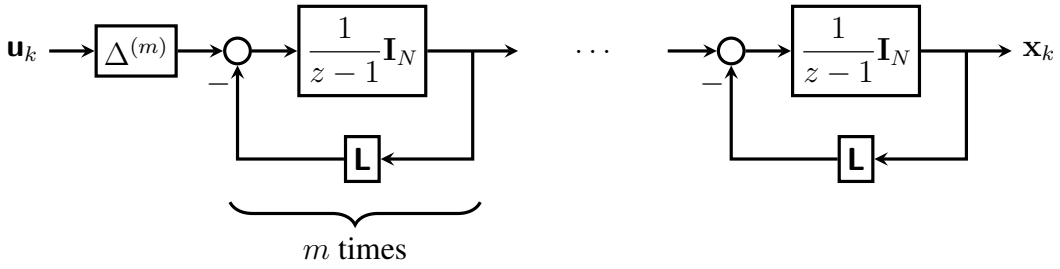
Signals with a known model (discrete time)

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

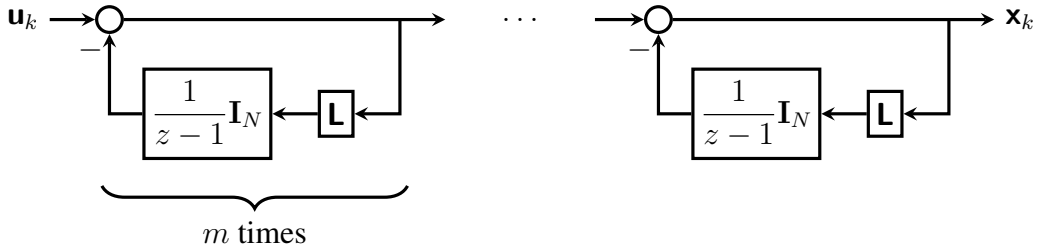
$i \in \{1, \ldots, N\}$. Dynamic average consensus algorithms can be designed which have zero steady-state error for such signals by placing the model of the input signals (that is, $d(z)$) in the feedback loop. Some common examples of models are

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

In this section, we will focus on dynamic average consensus algorithms which track degree $m-1$ polynomial reference signals with zero steady-state error.



(a) Dynamic average consensus algorithm (38) in [59] where $\Delta^{(m)} = (1 - z^{-1})^m$ is the m^{th} divided difference (see also [60] for a step size analysis). The performance 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) Dynamic average consensus algorithm (39), which is the algorithm in [44] cascaded in series m times. The estimate of the average is not delayed and the algorithm is numerically stable, but the tracking performance degrades when the communication graph is time-varying.

Figure 14: Block diagram of dynamic average consensus algorithms which 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.

Consider the dynamic average consensus algorithms in Figure 14. The transfer function of each algorithm has $m-1$ 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 14a are

$$x^i_{1,k+1} = x^i_{1,k} - \sum_{j=1}^{N} a_{ij}(x^i_{1,k} - x^j_{1,k}) + \Delta^{(m)}\mathsf{u}_k \tag{38a}$$

$$x^i_{2,k+1} = x^i_{2,k} - \sum_{j=1}^{N} a_{ij}(x^i_{2,k} - x^j_{2,k}) + x^i_{1,k} \tag{38b}$$

$$\vdots$$

$$x^i_{m,k+1} = x^i_{m,k} - \sum_{j=1}^{N} a_{ij}(x^i_{m,k} - x^j_{m,k}) + x^i_{m-1,k} \tag{38c}$$

$$x^i_k = x^i_{m,k}, \quad x^i_{\ell,0} \in \mathbb{R}, \quad \ell \in \{1, \dots, m\}, \quad i \in \{1, \dots, N\} \tag{38d}$$

where the m^{th} 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 $m \geq 2$ 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 14b, given by

$$p^i_{1,k+1} = p^i_{1,k} + \sum_{j=1}^{N} a_{ij}\big((\mathsf{u}^i_k - \mathsf{u}^j_k) - (p^i_{1,k} - p^j_{1,k})\big) \tag{39a}$$

$$p^i_{2,k+1} = p^i_{2,k} + \sum_{j=1}^{N} a_{ij}\big((\mathsf{u}^i_k - \mathsf{u}^j_k) - (p^i_{1,k} - p^j_{1,k}) - (p^i_{2,k} - p^j_{2,k})\big) \tag{39b}$$

$$\vdots$$

$$p^i_{m,k+1} = p^i_{m,k} + \sum_{j=1}^{N} a_{ij}\Big((\mathsf{u}^i_k - \mathsf{u}^j_k) - \sum_{\ell=1}^{m}(p^i_{\ell,k} - p^j_{\ell,k})\Big) \tag{39c}$$

$$x^i_k = \mathsf{u}^i_k - \sum_{\ell=1}^{m} p^i_{\ell,k}, \quad p^i_{\ell,0} \in \mathbb{R}, \quad \ell \in \{1, \dots, m\}, \quad i \in \{1, \dots, N\}, \tag{39d}$$

which tracks degree $m-1$ polynomial reference signals with zero steady-state error without delay. It should be noted, however, that we are assuming the communication graph is constant in order to use frequency domain arguments; while the output of the dynamic average consensus algorithm in Figure 14a is delayed, it also has nice tracking properties when the communication graph is time-varying whereas the dynamic average consensus algorithm in Figure 14b does not.

To track degree $m-1$ polynomial reference signals, each dynamic average consensus algorithm in Figure 14 cascades m dynamic average consensus algorithms, each with a pole at $z = 1$ in the feedback loop. The dynamic average consensus algorithm (15) is cascaded in Figure 14b, but any of the dynamic average consensus algorithms from the previous section could also be used. For example, the proportional-integral dynamic average consensus algorithm could be cascaded m

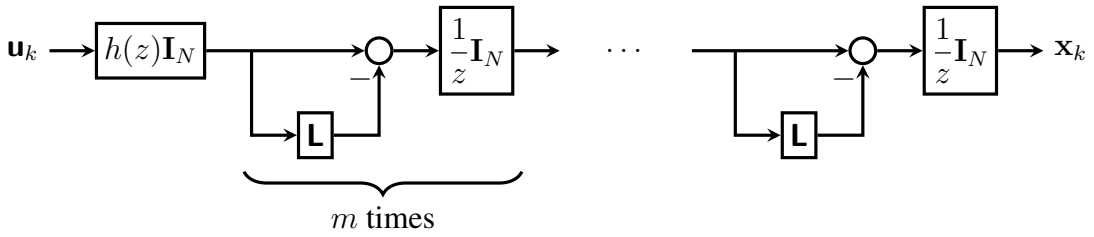


Figure 15: Feedforward dynamic average consensus algorithm for tracking the average of bandlimited reference signals. The prefilter $h(z)$ is applied to the reference signals *before* passing through the graph Laplacian. For an appropriately designed prefilter, the dynamic average consensus algorithm can track bandlimited reference signals with arbitrarily small steady-state error when using exact arithmetic (and small error for finite precision) [63].

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 $d(z) = d_1(z)\, d_2(z) \dots d_m(z)$. Then m dynamic average consensus algorithms can be cascaded where the i^{th} component contains the model $d_i(z)$ for $i = 1, \dots, m$. Alternatively, a single dynamic average consensus algorithm can be designed which contains the entire model $d(z)$. This approach using an internal model version of the proportional-integral dynamic average consensus algorithm is designed in [61] in both continuous and discrete time.

In many practical applications, the exact model of the reference signals is unknown. However, it is shown in [62] 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 [62] in place of the true frequency to design the feedback dynamic average consensus algorithm.

Bandlimited signals (discrete time)

In order 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 bandlimited. In this case, feedforward dynamic average consensus algorithm designs can be used to achieve arbitrarily small steady-state error.

For our discussion here, we assume that the reference signals are bandlimited with known cutoff frequency. In particular, let $\mathsf{U}^i(z)$ be the z-transform of the i^{th} reference signal $\{\mathsf{u}_k^i\}$. Then $\mathsf{U}^i(z)$ is bandlimited 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 15. 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

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

In order for the tracking error to be small, $h(z)$ should 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 $\|\mathbf{I} - \mathbf{L} - \mathbf{1}\mathbf{1}^\top/N\|_2 < 1$).

In particular, the prefilter should be 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 since this is not causal). An m-step filter can be obtained by cascading a one-step filter m times in series. In other words, let $h(z) = [zf(z)]^m$ where $f(z)$ is strictly proper and approximates unity in the passband. Since $f(z)$ must approximate unity in both magnitude *and* phase, a standard lowpass filter cannot be used. Instead, set

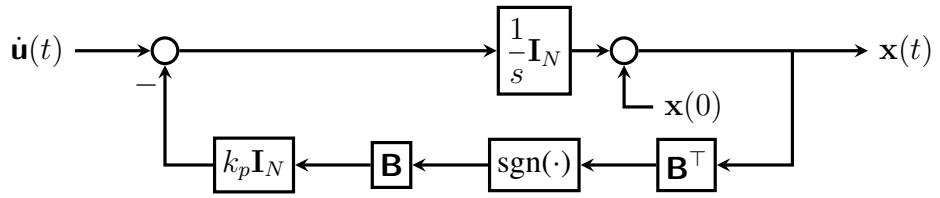
$$f(z) = 1 - g(z)/\lim_{z\to\infty} g(z)$$

where $g(z)$ is a proper highpass 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 $[0, \theta_c]$ (since $g(z)$ is highpass). Therefore, a prefilter $h(z)$ which approximates z^m in the passband can be designed using a standard highpass filter $g(z)$.

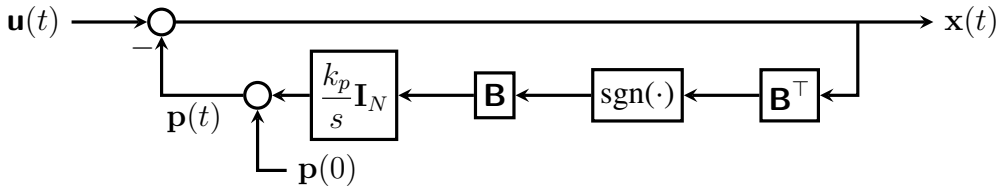
Using such a prefilter, [63] makes the error of the dynamic average consensus algorithm in Figure 15 arbitrarily small if (1) the graph is connected and balanced at each time step (in particular, it need not be constant), (2) $\mathbf{L}$ is scaled such that $\|\mathbf{I} - \mathbf{L} - \mathbf{1}\mathbf{1}^\top/N\|_2 < 1$, (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 error since rounding errors cause high frequency components in the reference signals.

Signals with bounded derivatives (continuous time)

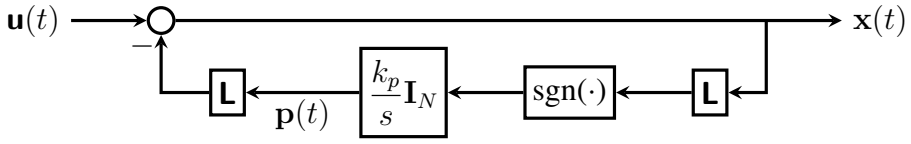
Stronger tracking results can be obtained using algorithms implemented in continuous time. Here, we present a continuous-time dynamic average consensus algorithm which is capable of tracking time-varying reference signals whose derivatives are bounded with zero error in *finite time*. For simplicity,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



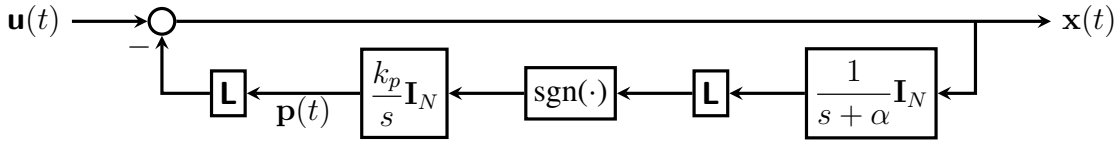
(a) Dynamic average consensus algorithm (40) which achieves perfect tracking in finite time and uses one-hop communication, but is not robust to initial conditions (that is, the steady-state error is zero only if $\mathbf{1}^\top\mathbf{x}(0) = \mathbf{1}^\top\mathbf{u}(0)$). Furthermore, the derivative of the reference signals is required; see [64].



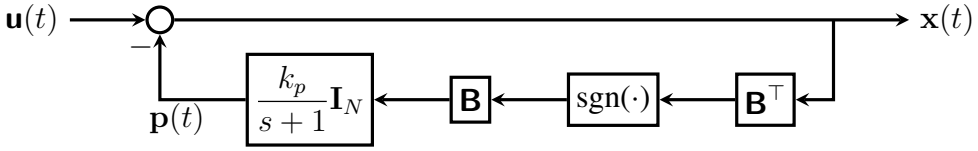
(b) Dynamic average consensus algorithm (41) which is equivalent to the algorithm in (a), although this form does not require the derivative of the reference signals. In this case, the requirement on the initial conditions is $\mathbf{1}^\top\mathbf{p}(0) = 0$.



(c) Dynamic average consensus algorithm (42) which 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 [65].



(d) Dynamic average consensus algorithm (43) which is robust to initial conditions and uses one-hop communication, but converges to zero error exponentially instead of in finite time; see [65].



(e) Dynamic average consensus algorithm (44) that is robust to initial conditions and uses one-hop communication, although the error converges to zero exponentially instead of in finite time; see [66].

Figure 16: 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.

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 proposed in [64] as

$$\dot{x}^i = \dot{\mathsf{u}}^i - k_p \sum_{j \in \mathcal{N}_{\text{out}}^i} \operatorname{sgn}(x^i(t) - x^j(t)), \qquad i \in \{1, \dots, N\}, \tag{40a}$$

$$\sum\nolimits_{i=1}^{N} x^i(0) = \sum\nolimits_{i=1}^{N} \mathsf{u}^i(0). \tag{40b}$$

The block diagram representation in Figure 16a indicates that this algorithm applies sgn in the feedback loop. Under the given assumptions, using a sliding mode argument, one can select the feedback gain k_p to guarantee zero error tracking in finite time provided that an upper bound γ of the form $\|\dot{\mathbf{u}}\|_{\text{ess}} = \gamma < \infty$ is known [64]. The dynamic consensus algorithm (40) can also be implemented without derivative information of the reference signals in an equivalent way as

$$\dot{p}^i = k_p \sum_{j \in \mathcal{N}_{\text{out}}^i} \operatorname{sgn}(x^i - x^j), \qquad \sum\nolimits_{j=1}^{N} p^j(0) = 0, \tag{41a}$$

$$x^i = \mathsf{u}^i - p^i, \qquad i \in \{1, \dots, N\}. \tag{41b}$$

The corresponding block diagram is shown in Figure 16b.

It is simple to see from the block diagram of Figure 16a why (40) 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 fixed by the dynamic average consensus algorithm in Figure 16c, given by

$$\dot{p}^i = k_p \operatorname{sgn}\Big(\sum_{j \in \mathcal{N}_{\text{out}}^i} (x^i - x^j) \Big), \qquad p^i(0) \in \mathbb{R}, \tag{42a}$$

$$x^i = \mathsf{u}^i - \sum_{j \in \mathcal{N}_{\text{out}}^i} (p^i - p^j), \qquad i \in \{1, \dots, N\}, \tag{42b}$$

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 do 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 only require one-hop communication, the dynamic average

consensus algorithm in Figure 16d, given by

$$\dot{q}^i = -\alpha\, q^i + x^i \tag{43a}$$

$$\dot{p}^i = k_p\, \mathrm{sgn}\Big(\sum_{j\in\mathcal{N}^i_{\text{out}}} (q^i - q^j)\Big), \tag{43b}$$

$$x^i = \mathsf{u}^i - \sum_{j\in\mathcal{N}^i_{\text{out}}} (p^i - p^j), \qquad p^i(0), q^i(0) \in \mathbb{R}, \qquad i \in \{1,\dots,N\}, \tag{43c}$$

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 [65].

Alternatively, under the given assumptions, a sliding mode-based dynamic consensus algorithm with zero error tracking which can be arbitrarily initialized is also proposed in [66] as

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

or equivalently,

$$\dot{p}^i = -p^i + k_p \sum_{j\in\mathcal{N}^i_{\text{out}}} \mathrm{sgn}(x^i - x^j), \qquad p^i(0) \in \mathbb{R}, \tag{44a}$$

$$x^i = \mathsf{u}^i - p^i, \qquad i \in \{1,\dots,N\}. \tag{44b}$$

However, this algorithm requires both the reference signals and their derivatives to be bounded with know values γ_1 and γ_2: $\|\mathbf{u}\|_{\text{ess}} = \gamma_1 < \infty$ and $\|\dot{\mathbf{u}}\|_{\text{ess}} = \gamma_2 < \infty$. These values are required to design the proper sliding mode gain k_p.

Conclusions

This paper has provided an overview of the state of the art on the available distributed algorithmic solutions to tackle the dynamic average consensus problem. We hope that the article has served the reader get an overview on the progress and intricacies of this topic, and helped appreciate the design trade-offs faced when balancing desirable properties 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 for a network system that the ability to track the average of time-varying reference signals using only local communication has, we expect the number and breadth of applications for dynamic average consensus algorithms to continue increasing. In particular, the recent emergence of interconnected scenarios where the reference signals come from another coordination algorithm, which in turn makes use itself of the variables obtained by the dynamic average consensus algorithm, is a fertile area where further research and applications might be developed in the coming years.

References

[1] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[2] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control: Collective group behavior through local interaction," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[3] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control*, ser. Communications and Control Engineering. Springer, 2008.

[4] W. Ren and Y. Cao, *Distributed Coordination of Multi-Agent Networks*, ser. Communications and Control Engineering. New York: Springer, 2011.

[5] P. Yang, R. Freeman, and K. Lynch, "Optimal information propagation in sensor networks," in *Proc. of the 2006 IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 3122–3127.

[6] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 710–724, 2008.

[7] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, 2008.

[8] 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.

[9] R. Aragüés, J. Cortés, and C. Sagüés, "Distributed consensus on robot networks for dynamically merging feature-based maps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 840–854, 2012.

[10] S. Das and J. M. F. Moura, "Distributed Kalman filtering with dynamic observations consensus," *IEEE Transactions on Signal Processing*, vol. 63, no. 17, pp. 4458–4473, Sep. 2015.

[11] F. Chen and W. Ren, "A connection between dynamic region-following formation control and distributed average tracking," *IEEE Transactions on Cybernetics*, 2018, to appear.

[12] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[13] W. Ren, "Multi-vehicle consensus with a time-varying reference state," *Systems and Control Letters*, vol. 56, no. 2, pp. 474–483, 2007.

[14] K. D. Listmann, M. V. Masalawala, and J. Adamy, "Consensus for formation control of nonholonomic mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 3886–3891.

[15] 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.

[16] S. Ghapani, S. Rahili, and W. Ren, "Distributed average tracking for second-order agents with nonlinear dynamics," in *American Control Conference*, 2016, pp. 4636–4641.

[17] S. S. Kia, J. Cortés, and S. Martínez, "Dynamic average consensus under limited control authority and privacy requirements," *International Journal on Robust and Nonlinear Control*, vol. 25, no. 13, pp. 1941–1966, 2015.

[18] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *IEEE Conf. on Decision and Control*, 2005, pp. 8179–8184.

[19] ——, "Kalman-consensus filter: Optimality, stability, and performance," in *IEEE Conf. on Decision and Control*, 2009, pp. 7036–7042.

[20] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, "Information weighted consensus filters and their application in distributed camera networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3112–3125, 2013.

[21] 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 Automatica Sinica*, vol. 40, no. 11, pp. 2632–2642, 2014.

[22] G. Wang, N. Li, and Y. Zhang, "Diffusion distributed Kalman filter over sensor networks without exchanging raw measurements," *Signal Processing*, vol. 132, pp. 1–7, 2017.

[23] W. Ren and U. M. Al-Saggaf, "Distributed Kalman-Bucy filter with embedded dynamic averaging algorithm," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–9, 2017.

[24] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[25] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Control and Optimization*, vol. 20, no. 3, pp. 1157–1170, 2009.

[26] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," in *IEEE Conf. on Decision and Control*, Orlando, Florida, 2011, pp. 3800–3805.

[27] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[28] J. Lu and C. Y. Tang, "Zero-gradient-sum algorithms for distributed convex optimization: the continuous-time case," *IEEE Transactions on Automatic Control*, vol. 57, no. 9, pp. 2348–2354, 2012.

[29] B. Gharesifard and J. Cortés, "Distributed continuous-time convex optimization on weight-balanced digraphs," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 781–786, 2014.

[30] 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, 2015.

[31] G. Qu and N. Li, "Accelerated distributed Nesterov gradient descent for convex and smooth functions," in *IEEE Conf. on Decision and Control*, 2017, pp. 2260–2267.

[32] Y. Nesterov, *Introductory lectures on convex optimization: a basic course*. Springer Science & Business Media, 2014, vol. 87.

[33] A. J. Wood, F. Wollenberg, and G. B. Sheble, *Power Generation, Operation and Control*, 3rd ed. New York: John Wiley, 2013.

[34] A. Cherukuri and J. Cortés, "Distributed generator coordination for initialization and anytime optimization in economic dispatch," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 226–237, 2015.

[35] R. Madan and S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 8, pp. 2185–2193, 2006.

[36] G. M. Heal, "Planning without prices," *Review of Economic Studies*, vol. 36, pp. 347–362, 1969.

[37] K. J. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Linear and Nonlinear Programming*. Stanford University Press, 1958.

[38] A. Cherukuri, B. Gharesifard, and J. Cortés, "Saddle-point dynamics: conditions for asymptotic stability of saddle points," *SIAM Journal on Control and Optimization*, vol. 55, no. 1, pp. 486–511, 2017.

[39] A. Cherukuri and J. Cortés, "Initialization-free distributed coordination for economic dispatch under varying loads and generator commitment," *Automatica*, vol. 74, pp. 183–193, 2016.

[40] S. S. Kia, "Distributed optimal in-network resource allocation algorithm design via a control theoretic approach," *Systems and Control Letters*, vol. 107, pp. 49–57, 2017.

[41] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, Nov. 1984.

[42] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Dynamic consensus on mobile networks," in *IFAC World Congress*, Prague, Czech Republic, Jul. 2005.

[43] B. Van Scoy, R. A. Freeman, and K. M. Lynch, "Design of robust dynamic average consensus estimators," in *IEEE Conf. on Decision and Control*, Dec. 2015, pp. 6269–6275.

[44] R. A. Freeman, P. Yang, and K. M. Lynch, "Stability and convergence properties of dynamic average consensus estimators," in *IEEE Conf. on Decision and Control*, San Diego, CA, Dec. 2006, pp. 398–403.

[45] S. S. Kia, J. Cortés, and S. Martínez, "Singularly perturbed filters for dynamic average consensus," in *European Control Conference*, Zürich, Switzerland, Jul. 2013, pp. 1758–1763.

[46] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[47] R. A. Freeman, T. R. Nelson, and K. M. Lynch, "A complete characterization of a class of robust linear average consensus protocols," in *American Control Conference*, 2010, pp. 3198–3203.

[48] E. Montijano, J. Montijano, C. Sagüés, and S. Martínez, "Robust discrete-time dynamic average consensus," *Automatica*, vol. 50, no. 12, pp. 3131–3138, 2014.

[49] B. V. Scoy, R. A. Freeman, and K. M. Lynch, "A fast robust nonlinear dynamic average consensus estimator in discrete time," in *Proc. of the 5th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, vol. 48, no. 22, 2015, pp. 191 – 196.

[50] B. Van Scoy, R. A. Freeman, and K. M. Lynch, "Optimal worst-case dynamic average consensus," in *American Control Conference*, Jul. 2015, pp. 5324–5329.

[51] ——, "Exploiting memory in dynamic average consensus," in *Proc. of the 53rd Annual Allerton Conf. on Comm., Control, and Computing*, Sep. 2015, pp. 258–265.

[52] B. N. Oreshkin, M. J. Coates, and M. G. Rabbat, "Optimization and analysis of distributed averaging with short node memory," *IEEE Transactions on Signal Processing*, vol. 58, no. 5, pp. 2850–2865, May 2010.

[53] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista, "Fast consensus by the alternating direction multipliers method," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5523–5537, Nov. 2011.

[54] E. Kokiopoulou and P. Frossard, "Polynomial filtering for fast convergence in distributed consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 342–354, Jan. 2009.

[55] Y. Yuan, J. Liu, R. M. Murray, and J. Gonçalves, "Decentralised minimal-time dynamic consensus," in *American Control Conference*, Jun. 2012, pp. 800–805.

[56] E. Montijano, J. I. Montijano, and C. Sagüés, "Chebyshev polynomials in distributed consensus applications," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 693–706, Feb. 2013.

[57] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "A systematic design process for internal model average consensus estimators," in *IEEE Conf. on Decision and Control*, Dec. 2013, pp. 5878–5883.

[58] ——, "Worst-case optimal average consensus estimators for robot swarms," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 3814–3819.

[59] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[60] E. Montijano, J. I. Montijano, C. Sagüés, and S. Martínez, "Step size analysis in discrete-time dynamic average consensus," in *American Control Conference*, Jun. 2014, pp. 5127–5132.

[61] H. Bai, R. A. Freeman, and K. M. Lynch, "Robust dynamic average consensus of time-varying inputs," in *IEEE Conf. on Decision and Control*, Atlanta, GA, Dec. 2010, pp. 3104–3109.

[62] H. Bai, "Adaptive motion coordination with an unknown reference velocity," in *American Control Conference*, Jul. 2015, pp. 5581–5586.

[63] B. V. 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 *IEEE Conf. on Decision and Control*, Dec. 2016, pp. 4284–4289.

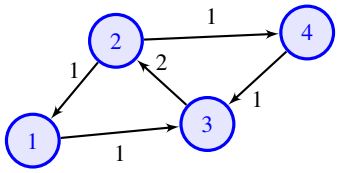
[64] F. Chen, Y. Cao, and W. Ren, "Distributed average tracking of multiple time-varying reference signals with bounded derivatives," *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3169–3174, 2012.

[65] J. George, R. A. Freeman, and K. M. Lynch, "Robust dynamic average consensus algorithm for signals with bounded derivatives," in *American Control Conference*, May 2017, pp. 352–357.

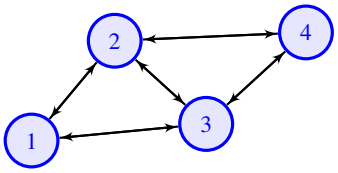
[66] S. Rahili and W. Ren, "Heterogeneous distributed average tracking using nonsmooth algorithms," in *American Control Conference*, Jul. 2017, pp. 691–696.

Sidebar 1: Basic Notions from Graph Theory

The communication network of a multiagent cooperative system can be modeled by a *directed graph* or simply a *digraph*. Here, we briefly review some basic concepts from graph theory following [S1]. 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 $(i, j) \in \mathcal{E}$ anytime $(j, i) \in \mathcal{E}$.



(a) Strongly connected, weight-balanced digraph:
$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, $\mathbf{L} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$.



(b) Connected graph with unit edge weights:
$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$, $\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$.

Figure 17: Examples of directed and undirected graphs.

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, $\mathrm{d}^{\text{out}}(i) = \sum_{j=1}^{N} a_{ji}$ and $\mathrm{d}^{\text{in}}(i) = \sum_{j=1}^{N} a_{ij}$. We 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 $\mathbf{1}_N^T \mathbf{L} = 0$. 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. We denote the eigenvalues of $\mathbf{L}$ by λ_i, $i \in \{1, \dots, N\}$, where $\lambda_1 = 0$ and $\Re(\lambda_i) \leq \Re(\lambda_j)$, for $i < j$. For strongly connected digraphs, one has $\mathrm{rank}(\mathbf{L}) = N - 1$. For strongly connected and weight-balanced digraphs, we denote the eigenvalues of $\mathrm{Sym}(\mathbf{L}) = (\mathbf{L} + \mathbf{L}^\top)/2$ by $\hat{\lambda}_1, \dots, \hat{\lambda}_N$, where $\hat{\lambda}_1 = 0$ and $\hat{\lambda}_i \leq \hat{\lambda}_j$, for $i < j$. For strongly connected and weight-balanced digraphs, we have

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

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

References

[S1] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*. Applied Mathematics Series, Princeton University Press, 2009. Available at http://www.coordinationbook.info.

Sidebar 2: Further Reading

There are many additional topics related to the dynamic average consensus problem that this article does not explore in detail; here we provide several interesting pointers for such topics for the interested reader.

A number of 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 [S2], additive input disturbances [S7], time-varying communication graphs [S17], and driving command saturation [17]. Variations of the dynamic average consensus problems explore scenarios where the algorithm design depends on the specific agent dynamics [66, S4, S3] or incorporates different agent roles, such as in leader-follower networks of mobile agents [S5, S6, S8].

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 [S9], it is shown in fact 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 which 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 [S10, S11]. 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 is referred to as the *push-sum* protocol and has been successfully applied to the static average consensus problem (see [S13, S14, S15, S16]). 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 only work if the time-varying graph remains weight-balanced, see [17, S28]. If communication failures caused by limited communication ranges or external events such as obstacle blocking destroy the weight-balanced character of the graph, it is still possible to solve the dynamic average consensus problem if the expected graph is balanced [S17]. Another set of works have 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 [S18] and optimizing the edge weights for fast consensus [5, S19].

References

[S2] H. Moradian and S. S. Kia, "Dynamic average consensus in the presence of communication delay over directed graph topologies," in *American Control Conference*, pp. 4663–4668, 2017.

[S3] S. Ghapania and W. Ren and 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.

[S4] F. Chen and G. Feng and L. Liu and W. Ren, "Distributed Average Tracking of Networked Euler-Lagrange Systems," in *IEEE Transactions on Automatic Control*, vol. 60, no. 2, pp. 547–552, 2015.

[S5] W. Ren, "Multi-vehicle consensus with a time-varying reference state," *Systems and Control Letters*, vol. 52, no. 2, pp. 474–483, 2007.

[S6] G. Shi and Y. Hong and K. H. Johansson, "Connectivity and set tracking of multi-agent systems guided by multiple moving leaders," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 663–676, 2012.

[S7] G. Shi and K. H. Johansson, "Robust consensus for continuous-time multi-agent dynamics," *SIAM Journal on Control and Optimization*, vol. 51, no. 5, pp. 3673–3691, 2013.

[S8] Z. Meng, D. V. Dimarogonas and K. H. Johansson, "Leader-follower coordinated tracking of multiple heterogeneous Lagrange systems using continuous control," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 739–745, 2014.

[S9] J. M. Hendrickx and J. N. Tsitsiklis, "Fundamental limitations for anonymous distributed systems with broadcast communications," in *Allerton Conference on Communication, Control, and Computing*, pp. 9–16, 2015.

[S10] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *European Journal of Control*, vol. 18, no. 6, pp. 539–557, 2012.

[S11] A. Rikos, T. Charalambous and C. N. Hadjicostis, "Distributed weight balancing over digraphs," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 2, pp. 190–201, 2014.

[S12] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *IEEE Symposium on Foundations of Computer Science*, (Washington, DC), pp. 482-491, 2003.

[S13] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *IEEE International Symposium on Information Theory Proceedings*, pp. 1753-1757, 2010.

[S14] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed matrix scaling and application to average consensus in directed graphs," *IEEE Transactions on Automatic Control*, vol. 58, no. 3, pp. 667–681, 2013.

[S15] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[S16] P. Rezaienia, B. Gharesifard, T. Linder, and B. Touri, "Push-sum on random graphs," https://arxiv.org/abs/1708.00915, 2017.

[S17] B. Van Scoy, R. A. Freeman, and K. M. Lynch, "Asymptotic mean ergodicity of average consensus estimators," in *American Control Conference*, 2014, pp. 4696–4701.

[S18] S. Kar and J.M.F. Moura, "Sensor Networks With Random Links: Topology Design for Distributed Consensus," in *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3315–3326, 2008.

[S19] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *IEEE Conf. on Decision and Control*, 2003, pp. 4997–5002.

Sidebar 3: Input-to-State Stability of LTI Systems

For a linear time-invariant system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad \mathbf{x} \in \mathbb{R}^n,\ \mathbf{u} \in \mathbb{R}^m, \tag{S1}$$

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

$$\mathbf{x}(t) = \mathrm{e}^{\mathbf{A}\,t}\,\mathbf{x}(0) + \int_0^t \mathrm{e}^{\mathbf{A}(t-\tau)}\,\mathbf{B}\,\mathbf{u}(\tau)\mathrm{d}\tau. \tag{S2}$$

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

$$\|\mathrm{e}^{\mathbf{A}\,t}\| \leq \kappa\,\mathrm{e}^{-\underline{\lambda}\,t}, \quad t \in \mathbb{R}_{\geq 0}, \tag{S3}$$

for some $\kappa, \underline{\lambda} \in \mathbb{R}_{>0}$, we can establish an upper bound on the norm of the trajectories of (S2) as follows

$$\begin{aligned}\|\mathbf{x}(t)\| &\leq \kappa\,\mathrm{e}^{-\underline{\lambda}\,t}\,\|\mathbf{x}(0)\| + \int_0^t \kappa\,\mathrm{e}^{-\underline{\lambda}\,(t-\tau)}\|\mathbf{B}\|\,\|\mathbf{u}(\tau)\|\mathrm{d}\tau,\\ &\leq \kappa\,\mathrm{e}^{-\underline{\lambda}\,t}\,\|\mathbf{x}(0)\| + \frac{\kappa\,\|\mathbf{B}\|}{\underline{\lambda}}\sup_{0\leq\tau\leq t}\|\mathbf{u}(\tau)\|, \qquad \forall t \in \mathbb{R}_{\geq 0}.\end{aligned} \tag{S4}$$

The bound here shows that the zero-input response decays to zero exponentially fast, while the zero-state response is bounded for every bounded input, indicating an input-to-state stability (ISS) behavior. It is worth noticing that the ultimate bound on the system state is proportional to the bound on the input.

Next, we comment on how to compute the parameters $\kappa, \underline{\lambda} \in \mathbb{R}_{>0}$ in (S3). Recall that [S20, Fact 11.15.5] for any matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, we can write

$$\|\mathrm{e}^{\mathbf{A}\,t}\| \leq \mathrm{e}^{\lambda_{\max}(\mathrm{Sym}(\mathbf{A}))\,t}. \quad \forall\, t \in \mathbb{R}_{\geq 0} \tag{S5}$$

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

$$\underline{\lambda} = -\lambda_{\max}(\mathrm{Sym}(\mathbf{A})), \quad \kappa = 1. \tag{S6}$$

A tighter exponential bound of

$$\underline{\lambda} = \lambda^\star, \quad \kappa = \sqrt{\sigma_{\max}(\mathbf{P}^\star)/\sigma_{\min}(\mathbf{P}^\star)}, \tag{S7}$$

can also be obtained, according to [S21, Proposition 5.5.33], from the convex linear matrix inequality optimization problem

$$(\lambda^\star, \mathbf{P}^\star) = \operatorname{argmin} \lambda \quad \text{s.t.} \tag{S8a}$$

$$\mathbf{P}\mathbf{A} + \mathbf{A}^\top \mathbf{P} \leq -2\,\lambda\,\mathbf{P}, \quad \mathbf{P} > 0, \quad \lambda > 0. \tag{S8b}$$

Similarly, the state of the discrete-time linear time-invariant system

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \quad \mathbf{x}_k \in \mathbb{R}^n,\ \mathbf{u}_k \in \mathbb{R}^m \tag{S9}$$

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

$$\|\mathbf{x}_k\| \leq \sqrt{\frac{\sigma_{\max}(\mathbf{P})}{\sigma_{\min}(\mathbf{P})}}\Big(\rho^k \|\mathbf{x}_0\| + \frac{1-\rho^k}{1-\rho}\|\mathbf{B}\| \sup_{0 \leq m < k} \|\mathbf{u}_m\|\Big) \tag{S10}$$

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

$$\mathbf{A}^\top \mathbf{P} \mathbf{A} - \rho^2\,\mathbf{P} \leq 0, \quad \mathbf{P} > 0, \quad \rho \geq 0. \tag{S11}$$

References

[S20] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear System Theory*. Springer-Verlag Berlin Heidelberg, 2005.

[S21] D. Hinrichsen and A. J. Pritchard, *Mathematical Systems Theory I: Modeling, State Space Analysis, Stability and Robustness*. Princeton University Press, 2005.

Sidebar 4: Euler Discretizations of Continuous-Time Dynamic Average Consensus Algorithms

The continuous-time algorithms described in the paper 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 stepsize. To illustrate the discussion, we develop this approach for the algorithm (23) over a connected graph topology. The discussion below can also be extended to include iterative forms of the other continuous-time algorithms studied in the paper. Using the Euler discretization in the algorithm (23) leads to

$$v^i(k+1) = v^i(k) + \delta\alpha\beta \sum\nolimits_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)), \tag{S12a}$$

$$x^i(k+1) = x^i(k) + \Delta \mathsf{u}^i(k) - \delta\alpha(x^i(k) - \mathsf{u}^i(k)) - \delta\beta \sum\nolimits_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)) - \delta v^i(k), \tag{S12b}$$

where $\Delta \mathsf{u}^i(k) = \mathsf{u}^i(k+1) - \mathsf{u}^i(k)$. To implement this iterative form at each timestep k we need to have access to the future value of the reference input at timestep $k+1$. Such a requirement is not practical when the reference input is sampled from a physical process or is a result of an another online algorithm. One could circumvent this requirement using a backward Euler discretization but the resulting algorithm will track the reference dynamic average with one step delay. A practical solution which avoids requiring the future values of the reference input is obtained by introducing an intermediate variable $z^i(k) = x^i(k) - \mathsf{u}^i(k)$ and representing the iterative algorithm (S12) in the form

$$v^i(k+1) = v^i(k) + \delta\alpha\beta \sum\nolimits_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)), \tag{S13a}$$

$$z^i(k+1) = z^i(k) - \delta\alpha z^i(k) - \delta\beta \sum\nolimits_{j=1}^{N} a_{ij}(x^i(k) - x^j(k)) - \delta v^i(k), \tag{S13b}$$

$$x^i(k) = z^i(k) + \mathsf{u}^i(k), \tag{S13c}$$

for $i \in \{1, \dots, N\}$. Algorithm (S13) is then the form that should be implement.

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

Lemma 0.1 (Admissible stepsize for Euler discretized form of LTI 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

$$\mathbf{x}(k+1) = (\mathbf{I} + \delta\mathbf{A})\,\mathbf{x}(k) + \delta\mathbf{B}\,\mathbf{u}(k), \quad k \in \mathbb{Z}_{\geq 0}, \tag{S14}$$

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 descretizetion stepsize. 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 input signal be bounded, $\|\Delta\mathbf{u}\| < \rho < \infty$. Then, for any $\delta \in (0, \bar{d})$ where

$$\bar{d} = \min\Big\{ -2\,\frac{\mathrm{Re}(\mu_i)}{|\mu_i|^2} \Big\}_{i=1}^{n} \tag{S15}$$

the eigenvalues of $(\mathbf{I} + \delta\mathbf{A})$ *are all located inside the until circle in complex plane. Moreover, starting from any* $\mathbf{x}(0) \in \mathbb{R}^n$*, the trajectories of* (S14) *satisfy*

$$\lim_{k\to\infty} \|\mathbf{x}(k+1)\| \leq \frac{\kappa\,\rho\,\|\mathbf{B}\|}{1-\omega}, \tag{S16}$$

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 LMI optimization problem (see [S22, Theorem 23.3] for details)

$$\begin{aligned}(\omega, \kappa, \mathbf{Q}) = \operatorname{argmin} \omega^2, \quad \text{subject to} \\ \frac{1}{\kappa}\,\mathbf{I} \leq \mathbf{Q} \leq \mathbf{I}, \quad 0 < \omega^2 < 1, \quad \kappa > 1, \\ (\mathbf{I} + \delta\mathbf{A})^\top \mathbf{Q}\,(\mathbf{I} + \delta\mathbf{A}) - \mathbf{Q} \leq -(1-\omega^2)\,\mathbf{I}.\end{aligned} \tag{S17}$$

Building on Lemma 0.1, the next result characterizes the admissible discretization stepsize for the algorithm (S13) and its ultimate tracking behavior.

Theorem 0.9 (Convergence of (S13) over connected graphs [17])**.** *Let* $\mathcal{G}$ *be a connected graph. Assume that the differences of the inputs of the network satisfy* $\|(\mathbf{I} - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top)\,\Delta\mathbf{u}(k)\| = \gamma < \infty$*. Then, for any* $\alpha, \beta > 0$*, the algorithm* (S13) *over* $\mathcal{G}$ *initialized at* $z^i(0) \in \mathbb{R}$ *and* $v^i(0) \in \mathbb{R}$ *such that* $\sum_{i=1}^N v^i(0) = 0$ *has bounded trajectories that satisfy*

$$\lim_{k\to\infty} \left|x^i(k) - \mathsf{u}^{avg}(k)\right| \leq \frac{\delta\,\kappa\,\gamma}{1-\omega}, \quad i \in \{1,\dots,N\} \tag{S18}$$

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* $\bar{\kappa} \in \mathbb{R}_{>0}$ *satisfy* $\|\mathbf{I} - \delta\,\beta\,\mathbf{R}^\top\mathbf{L}\mathbf{R}\|^k \leq \bar{\kappa}\,\bar{\omega}^k$*,* $k \in \mathbf{Z}_{\geq 0}$*.*

Note that the characterization of the stepsize requires on knowledge of the largest eigenvalue λ_N of the Laplacian. Since this is not readily available to the network unless dedicated distributed algorithms are introduced to compute it, [17] provides the sufficient characterization $\delta \in (0, \min\{\alpha^{-1}, \beta^{-1}(\mathsf{d}_{\text{out}}^{\max})^{-1}\})$ along with the ultimate tracking bound

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

References

[S22] W. J. Rugh, *Linear Systems Theory*. Prentice Hall: Englewood Cliffs, NJ, 1993.

Sidebar 5: Dynamic Average Consensus Algorithms with Continuous-Time Evolution and Discrete-Time Communication

We discuss here an alternative to the discretization route explained in "Sidebar 3: Euler discretizations of continuous-time dynamic average consensus algorithms" to produce implementable strategies from the continuous-time algorithms described in the paper. 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 every growing computing capabilities of modern embedded processors and computers) whereas communication is a process that still requires proper acknowledgment of its discrete-time nature. Our basic idea is to opportunistically trigger, based on the network state, the times when information sharing among agents should take place and allow individual agents to determine these autonomously. This has the potential to result in more efficient algorithm implementations as performing communication usually requires more energy than computation [S23]. In addition, the use of fixed communication stepsizes 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 [S24, S25] 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 recent surveys [S26, S27] describe how these ideas can be employed to design event-triggered communication laws for static average consensus.

Motivated by these observations, [S28] investigates a discrete-time communication implementation of the continuous-time algorithm (23) for dynamic average consensus. Under this strategy the algorithm becomes

$$\dot{v}^i = \alpha\beta \sum\nolimits_{j=1}^{N} \mathsf{a}_{ij}(\hat{x}^i - \hat{x}^j), \tag{S19a}$$

$$\dot{x}^i = \dot{\mathsf{u}}^i - \alpha(x^i - \mathsf{u}^i) - \beta \sum\nolimits_{j=1}^{N} \mathsf{a}_{ij}(\hat{x}^i - \hat{x}^j) - v^i, \tag{S19b}$$

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: (a) triggers need to be distributed, so that agents can check them with the information available to them from their out-neighbors, (b) 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 (c) they have to ensure the network achieves dynamic average consensus even though agents operate with outdated information while inputs are changing with time.

Consider the following event-triggered communication law [S28]: 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

$$t_{k+1}^i = \text{argmax}\{t \in [t_k^i, \infty) \,|\, |x^i(t_k^i) - x^i(t)| \leq \epsilon_i\}. \tag{S20}$$

Here, $\epsilon_i \in \mathbb{R}_{>0}$ is a constant value which each agent chooses locally to control its inter-event times and avoid Zeno behavior. Specifically, one can show that the inter-execution times of each agent $i \in \{1, \ldots, N\}$ employing (S20) are lower bounded by

$$\tau^i = \frac{1}{\alpha} \ln\Big(1 + \frac{\alpha\epsilon_i}{c^i}\Big), \tag{S21}$$

where c^i and η are positive real numbers that depend on the initial conditions and network parameters (we omit here for simplicity their specific form, but the interested reader is referred to [S28] for explicit expressions). The lower bound (S21) shows that for a positive non-zero ϵ^i, the inter-execution times are bounded away from zero and we have the guarantees that for networks with finite number of agents the implementation of the algorithm (S19) with the communication trigger law (S20) is Zeno free. The following result formally describes the convergence behavior of the algorithm (S19) under (S20) When the interaction topology is modeled by a strongly connected and weight-balanced digraph.

Theorem 0.10 (Convergence of (S19) over strongly connected and weight-balanced digraph with asynchronous distributed event-triggered communication [S28])**.** *Assume the reference signals satisfy $|\dot{\mathsf{u}}^i|_{ess} = \kappa^i < \infty$, for $i \in \{1, \ldots, N\}$, and $\|\mathbf{\Pi}_N \dot{\mathbf{u}}\|_{ess} = \gamma < \infty$. Let the communication topology be a weight-balanced and strongly connected digraph $\mathcal{G}$. For any $\alpha, \beta \in \mathbb{R}_{>0}$, the algorithm* (S19) *over $\mathcal{G}$ starting from $x^i(0) \in \mathbb{R}$ and $v^i(0) \in \mathbb{R}$ with $\sum_{i=1}^N v^i(0) = 0$, where each agent $i \in \{1, \ldots, N\}$ 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* (S20) *with $\boldsymbol{\epsilon} \in \mathbb{R}_{>0}^N$, satisfies*

$$\limsup_{t\to\infty} \Big| x^i(t) - \mathsf{u}^{avg}(t) \Big| \leq \frac{\gamma + \beta \|\mathbf{L}\| \, \|\boldsymbol{\epsilon}\|}{\beta \hat{\lambda}_2}, \tag{S22}$$

for $i \in \{1, \ldots, N\}$ with an exponential rate of convergence of $\min\{\alpha, \beta\hat{\lambda}_2\}$. Furthermore, the inter-execution times of agent $i \in \{1, \ldots, N\}$ are lower bounded by (S21)*.*

The expected trade-off between the desire for longer inter-event time and the adverse effect on systems convergence and performance is captured in (S21) and (S22). The lower bound τ^i in (S21) 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, one 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 (S22) results in smaller inter-event times. Given that the rate of convergence of (S19) under (S20) 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 allow us to identify an alternative event-triggered communication law which, compared to (S20), has a longer inter-event time but similar dynamic average tracking performance. Now, according to [S28], the communication times $\{t_k^i\}_{k\in\mathbb{N}}$ are determined by

$$t_{k+1}^i = \operatorname{argmax}\{t \in [t_k^i, \infty) \mid |\hat{x}^i(t) - x^i(t)|^2 \leq \frac{1}{4\mathsf{d}_{\text{out}}^i}\sum_{j=1}^N \mathsf{a}_{ij}(t)|\hat{x}^i(t) - \hat{x}^j(t)|^2 + \frac{1}{4\mathsf{d}_{\text{out}}^i}\epsilon_i^2\}, \tag{S23}$$

Compared to (S20), the extra term $\frac{1}{4\mathsf{d}_{\text{out}}^i}\sum_{j=1}^N \mathsf{a}_{ij}(t)|\hat{x}^i(t)-\hat{x}^j(t)|^2$ in the communication law (S23) allows agents to have longer inter-event times. Formally, one can show that the inter-execution times of agent $i \in \{1, \dots, N\}$ implementing (S23) are lower bounded by

$$\tau^i = \frac{1}{\alpha}\ln\Big(1 + \frac{\alpha\epsilon_i}{2\bar{c}^i\sqrt{\mathsf{d}_{\text{out}}^i}}\Big), \tag{S24}$$

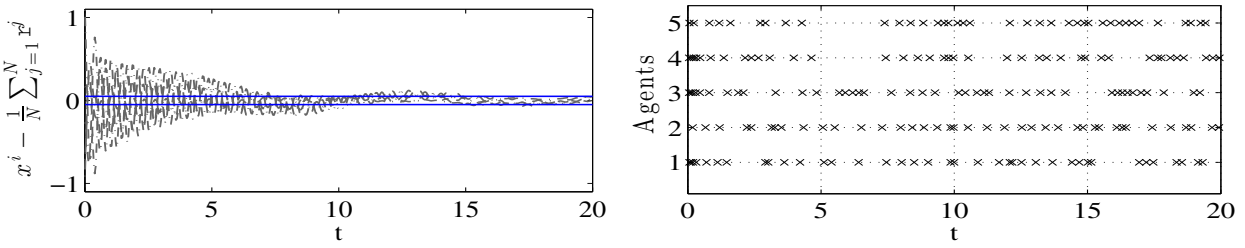
for positive constants $\bar{c}^i$, see [S28] for explicit expressions. Numerical examples in [S28] show that the implementation of (S23) for connected graphs results in inter-event times longer than the ones of the event-triggered law (S20). Figure 18 shows one of those examples. The following result formally describes the convergence behavior of the algorithm (S19) under (S23) over connected graphs.

Theorem 0.11 (Convergence of (S19) over connected graphs with asynchronous distributed event-triggered communication [S28])**.** *Assume the reference signals satisfy $|\dot{\mathsf{u}}^i|_{ess} = \kappa^i < \infty$, for $i \in \{1, \dots, N\}$, and $\|\mathbf{\Pi}_N\dot{\mathbf{u}}\|_{ess} = \gamma < \infty$. Let the communication topology be a connected graph $\mathcal{G}$. For any $\alpha, \beta \in \mathbb{R}_{>0}$, the algorithm* (S19) *over $\mathcal{G}$ starting from $x^i(0) \in \mathbb{R}$ and $v^i(0) \in \mathbb{R}$ with $\sum_{i=1}^N v^i(0) = 0$, where 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* (S23) *with $\boldsymbol{\epsilon} \in \mathbb{R}_{>0}$, satisfies*

$$\limsup_{t\to\infty}\Big|x^i(t) - \mathsf{u}^{avg}(t)\Big| \leq \frac{\gamma}{\beta\lambda_2} + \sqrt{\Big(\frac{\gamma}{\beta\lambda_2}\Big)^2 + \frac{\|\boldsymbol{\epsilon}\|^2}{2\lambda_2}},$$

for $i \in \{1, \dots, N\}$. Furthermore, the inter-execution times of agent $i \in \{1, \dots, N\}$ are lower bounded by (S24)*.*

The results reported here can also be extended for time-varying, jointly connected graphs, see [S28] for a complete exposition.



(a) (b)

Figure 18: Comparison between the event-triggered algorithm (S19) employing the event-triggered communication law (S23) and the Euler discretized implementation of algorithm (23) as described in (S13) with fixed stepsize [S28]. Both of these algorithms use $\alpha = 1$ and $\beta = 4$. The network is a weight-balanced digraph of 5 agents with unit weights. The inputs are $\mathsf{r}^1(t) = 0.5\sin(0.8t)$, $\mathsf{r}^2(t) = 0.5\sin(0.7t) + 0.5\cos(0.6t)$, $\mathsf{r}^3(t) = \sin(0.2t) + 1$, $\mathsf{r}^4(t) = \text{atan}(0.5t)$, $\mathsf{r}^5(t) = 0.1\cos(2t)$. In plot (a), the black (resp. gray) lines correspond to the tracking error of the event-triggered algorithm (S19) employing event-triggered law (S23) with $\epsilon_i/(2\sqrt{\mathsf{d}^i_{\text{out}}}) = 0.1$ (resp. the Euler discretized algorithm (S13) with fixed stepsize $\delta = 0.12$). Recall from "Sidebar 4:Euler Discretizations of Continuous-Time Dynamic Average Consensus Algorithms" that convergence for algorithm (S13) is guaranteed if $\delta \in (0, \min\{\alpha^{-1}, \beta^{-1}(\mathsf{d}^{\text{out}}_{\max})^{-1}\})$, which for this example results in $\delta \in (0, 0.125)$. The horizontal blue lines show the ± 0.05 error bound for reference. Plot (b) shows the communication times of each agent using the event-triggered strategy. As seen in plot (a), both these algorithms exibit comparable tracking performance. The number of times that agents $\{1, 2, 3, 4, 5\}$ communicate in the time interval $[0, 20]$ is $(39, 40, 42, 40, 39)$, respectively, when implementing event-triggered communication (S23). This is significantly less than the communication used by each agent in the Euler discretized algorithm (S13) ($20/0.12 \simeq 166$ rounds).

References

[S23] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. John Wiley: New York, 2005.

[S24] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *IEEE Conf. on Decision and Control*, pp. 3270–3285, Maui, HI, 2012.

[S25] L. Hetel, C. Fiter, H. Omran, A. Seuret, E. Fridman, J. Richard, and S. I. Niculescu, "Recent developments on the stability of systems with aperiodic sampling: an overview," *Automatica*, vol. 76, pp. 309–335, 2017.

[S26] C. Nowzari, E. Garc'ía, and J. Cortés, "Event-triggered control and communication of network systems for multi-agent consensus," `https://arxiv.org/abs/1712.00429`, 2017.

[S27] L.Ding, Q. L. Han X. Ge, and X. M. Zhang, "An overview of recent advances in event-triggered consensus of multiagent systems," *IEEE Transactions on Cybernetics*, 2018, to appear.

[S28] 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, 2015.

Authors Information

Solmaz S. Kia is an Assistant Professor of Mechanical and Aerospace Engineering at the University of California, Irvine, CA, USA. She obtained her Ph.D. degree in Mechanical and Aerospace Engineering from UCI, in 2009, and her M.Sc. and B.Sc. in Aerospace Engineering from the Sharif University of Technology, Iran, in 2004 and 2001, respectively. She was a senior research engineer at SySense Inc., El Segundo, CA from Jun. 2009 to Sep. 2010. She held postdoctoral positions in the Department of Mechanical and Aerospace Engineering at the UC San Diego and UCI. She was a recipient of UC president's postdoctoral fellowship in 2012-2014 and NSF 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, WI, USA. He received the Ph.D. degree in Electrical Engineering from Northwestern University in 2017, and B.S. and M.S. degrees in Applied Mathematics along with a B.S.E. in Electrical Engineering from the University of Akron in 2012. His research interests include distributed algorithms for multi-agent systems, and the analysis and design of optimization algorithms.

Jorge Cortés (M'02-SM'06-F'14) is a Professor of Mechanical and Aerospace Engineering at the University of California, San Diego, CA, USA. He received the Licenciatura degree in mathematics from Universidad de Zaragoza, Zaragoza, Spain, in 1997, and the Ph.D. degree in engineering mathematics from Universidad Carlos III de Madrid, Madrid, Spain, in 2001. He held postdoctoral positions with the University of Twente, Twente, The Netherlands, and the University of Illinois at Urbana-Champaign, Urbana, IL, USA. He was an Assistant Professor with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA, USA, from 2004 to 2007. He is the author of Geometric, Control and Numerical Aspects of Nonholonomic Systems (Springer-Verlag, 2002) and co-author (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, multi-agent coordination in robotics, power systems, and neuroscience, geometric and distributed optimization, nonsmooth analysis, and geometric mechanics and control.

Randy Freeman is a Professor of Electrical Engineering and Computer Science at Northwestern University, Evanston, IL, USA. He received the Ph.D. degree in electrical engineering from the University of California at Santa Barbara, Santa Barbara, CA, USA, in 1995. He has been Associate Editor of the IEEE Transactions on Automatic Control. His research interests include nonlinear systems, distributed control, multiagent systems, robust control, optimal control, and oscillator synchronization.Prof. Freeman 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 Lynch (S'90-M'96-SM'05-F'10) is a Professor and the Chair of the Mechanical Engineering Department, Northwestern University, Evanston, IL, USA. He received the B.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 1989, and the Ph.D. degree in robotics from Carnegie Mellon University, Pittsburgh, PA, USA, 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 (M'02-SM'07-F'18) is a Professor of Mechanical and Aerospace Engineering at the University of California, San Diego, CA, USA. 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, Urbana-Champaign during 2004, and at the Center for Control, Dynamical systems and Computation (CCDC) of the University of California, Santa Barbara during 2005. In a broad sense, her main research interests include the control of networked systems, multi-agent 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 NSF CAREER Award in 2007. For the paper "Motion coordination with Distributed Information," co-authored with Jorge Cortés and Francesco Bullo, she received the 2008 Control Systems Magazine Outstanding Paper Award. She has served on the editorial boards of the European Journal of Control (2011-2013), and currently serves on the editorial board of the Journal of Geometric Mechanics and IEEE Transactions on Control of Networked Systems.