Accelerated Multi-Stage Discrete Time Dynamic Average Consensus

Eduardo Sebastián, Eduardo Montijano, Carlos Sagüés, Mauro Franceschelli and Andrea Gasparri

***Abstract*— This paper presents a novel solution for the discrete time dynamic average consensus problem. Given a set of time-varying input signals over the nodes of an undirected graph, the proposed algorithm tracks, at each node, the input signals' average. The algorithm is based on a sequence of consensus stages combined with a second order diffusive protocol. The former overcomes the need of k-th order differences of the inputs and conservation of the network state average, while the latter overcomes the trade-off between speed and accuracy of the consensus stages by just storing the previous estimate at each node. The result is a protocol that is fast, arbitrarily accurate, and robust against input noises and initializations. The protocol is extended to an asynchronous and randomized version that follows a gossiping scheme that is robust against potential delays and packet losses. We study the convergence properties of the algorithms and validate them via simulations.**

***Index Terms*— Consensus, distributed control, estimation, sensor networks**

I. Introduction

The problem of consensus in control theory [1] consists of finding a protocol such that a set of nodes in a network agree on the value of a certain quantity of interest. In particular, the discrete time dynamic average consensus [2] is interesting because the tracked signals usually evolve with time, and protocols are implemented in computing units that work in discrete steps. Despite the existing literature, current solutions still suffer from some of the following issues: (i) integral and difference input terms are not robust against input and initialization noise, or changes of network size; (ii) trade-off between convergence speed and steady-state accuracy; (iii) absence of robustness against packet losses and communication delays. These aspects affect the applicability of consensus in real-world scenarios such as smart grids [3] or sensor networks [4].

This work was supported by the Spanish projects PID2021-125514NB-I00, PID2021-124137OB-I00, TED2021-130224B-I00 funded by MCIN/AEI/10.13039/501100011033, by ERDF A way of making Europe and by the European Union NextGenerationEU/PRTR, by the Gobierno de Aragón under Project DGA T45-23R, and by Spanish grant FPU19-05700.

E. Sebastián, E. Montijano and C. Sagüés are with the Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Spain (email:esebastian@unizar.es, emonti@unizar.es, csagues@unizar.es). M. Franceschelli is with the Department of Electrical and Electronic Engineering, University of Cagliari, Italy (email: mauro.franceschelli@unica.it). A. Gasparri is with the Department of Engineering, Roma Tre University, Italy (email:gasparri@dia.uniroma3.it).

To overcome these issues, we propose a novel algorithm that combines a multi-stage consensus protocol and a second order diffusion method, along with its asynchronous and randomized version.

Several solutions have been proposed to improve the capabilities of discrete time dynamic average consensus algorithms. For instance, some works achieve an arbitrarily small steady-state error by exploiting the k-th differences of the input signal [5], [6]. This is problematic when the inputs are noisy since noise breaks the boundedness of the input signal. An alternative is to rely only on the input signal, achieving an arbitrarily small steady-state error by concatenating a cascade of consensus filters [7], [8], as we do in this work. Besides, this leads to other desirable properties like robustness against non-averaged initializations and changes in the network size. The counterpart is a slow convergence. Regarding robustness, there exist continuous-time algorithms [9]–[12] that address initialization issues and time-varying networks. Our algorithm achieves the same robustness but in the desired discrete-time setting.

The issue of slow convergence has been considered from two main perspectives. Ghosh et al. [13] present second order diffusion methods, which significantly speed up the convergence [14] by increasing the memory requirements with the previous estimate. These works are for static problems, while our work deals with the dynamic consensus problem. On the other hand, polynomial filters [15] consider a sequence of consensus iterations as the evaluation of a polynomial. For instance, Montijano et al. [6], [16] analyze Chebyshev polynomials, proving that they significantly increase the convergence speed. However, the k-th order differences of the input signal are exploited. Thus, this work opts for a second order method to speed up convergence. Closely related to our paper, Van Scoy et al. [17] present a fast and robust discrete time dynamic average consensus estimator. Compared to them, our proposal does not need bounded inputs while achieving the same robustness against initialization errors and accelerated convergence. Beyond consensus, compression techniques [18] or decomposition principles [19] can be used to reduce the dimensionality of the data and computational cost.

Finally, regarding robustness against potential communication delays and packet losses, many applications deal with time-varying networks, where nodes connect and disconnect depending on events exogenous to the consensus protocol. To account for this, gossip algorithms [20], [21] propose consensus protocols computed at random asynchronous instants

and neighbors. Other works either propose reformulations over linear proportional consensus protocols against packet drops [22] or continuous-time protocols against delays [23]. This motivates our second proposed algorithm, which is an accelerated version of a discrete time dynamic average consensus protocol that runs asynchronously and with randomized communication links.

This paper builds from the works by Franceschelli and Gasparri [7], [8]. They present a multi-stage discrete time dynamic average consensus filter that is fully distributed, does not use the k-th order differences of the inputs signals, and is robust against non-averaged nodes' initialization. Besides, the steady-state error can be arbitrarily reduced by increasing the number of stages. The main limitation is the slow convergence, so there is a trade-off between accuracy and speed. To solve this issue, we propose a second order method to accelerate the convergence, which permits to either increase the number of stages (and reduce the steady-state error) with the same convergence speed of the original filter, or speed up the convergence for the same number of stages while maintaining the steady-state error. Our main contribution is an acceleration method to speed up (i) the multi-stage discrete time dynamic average consensus protocol, which yields to quick, accurate and robust tracking of the dynamic average, and (ii) the asynchronous and randomized version of the former.

II. Preliminaries

The system[1] under study is a network composed by $N > 1$ nodes. The network is described by an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \dots, i, \dots, N\}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Nodes i and j can exchange information if and only if $(i, j) \in \mathcal{E}$, which implies that $(j, i) \in \mathcal{E}$. $\mathcal{N}_i = \{j | (j, i) \in \mathcal{E}\}$ is the neighborhood of node i. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ associated to $\mathcal{G}$ is such that $\mathbf{A}_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and 0 otherwise. $\mathbf{A}_{ii} = 0$ always because we do not allow self-loops. The degree matrix $\mathbf{D}$ associated to $\mathcal{G}$ is such that $\mathbf{D}_{ij} = \mathsf{car}(\mathcal{N}_i)\ \forall i = j$ and 0 otherwise. The Laplacian matrix associated to $\mathcal{G}$ is $\mathbf{L} = \mathbf{D} - \mathbf{A}$. For undirected graphs, it holds that $\mathbf{L}$ is symmetric and has real eigenvalues. The sorted eigenvalues of $\mathbf{L}$ are $\lambda_1(\mathbf{L}) < \ldots \leq \lambda_N(\mathbf{L}) \leq 2D_{\max}$, where the relation $D_{\max} = \max(\{\mathbf{D}_{ii}\}_{i=1}^N)$ holds. The second smallest eigenvalue, $\lambda_2(\mathbf{L})$, is called *algebraic connectivity*. We denote $\mathbf{v}_{r,i}(\mathbf{L}), \mathbf{v}_{l,i}(\mathbf{L})$ the right and left eigenvectors of $\mathbf{L}$ associated to the i-th eigenvalue.

A. Second Order Diffusion Methods

The linear (first-order) discrete time *static* average consensus algorithm ([24]) is

$$x_i(k+1) = \mathbf{W}_{ii} x_i(k) + \textstyle\sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij} x_j(k), \qquad (1)$$

with $x_i(0)$ the initial condition and $\mathbf{W} \in \mathbb{R}^{N \times N}$ a weighted matrix. In matrix form, (1) leads to

$$\mathbf{x}(k+1) = \mathbf{W}\mathbf{x}(k). \qquad (2)$$

If $\mathbf{W}$ is doubly stochastic, i.e., $\mathbf{1}^T\mathbf{W} = \mathbf{1}^T$ and $\mathbf{W}\mathbf{1} = \mathbf{1}$, then it is known that the protocol in Eq. (2) converges to the average of the initial states $\bar{x}_i(0) = (1/N)\sum_{i=1}^N x_i(0)$.

Without loss of generality, we consider $\mathbf{W} = \mathbf{I} - \epsilon\mathbf{L}$. The convergence speed can be too slow, specially in networks with a small algebraic connectivity ($\lambda_2(\mathbf{W}) = 1 - \epsilon\lambda_2(\mathbf{L})$). To accelerate convergence, Ghosh et al. ([13]) proposed a second-order modification:

$$\mathbf{x}(k+1) = \gamma\mathbf{W}\mathbf{x}(k) + (1-\gamma)\mathbf{x}(k-1). \qquad (3)$$

The properties of the protocol in Eq. (3) are well-studied (e.g., [13]). In particular, protocol (3) improves the convergence speed of (2) if $\gamma \in (1, 2)$.

B. Multi-Stage Discrete Time Dynamic Average Consensus

The concepts in subsection II-A apply to the static consensus problem, but this work deals with a dynamic consensus problem. Node i has an input $r_i(k) \in \mathbb{R}$ and an estimate $x_i(k) \in \mathbb{R}$ associated to a quantity of interest $r(k) \in \mathbb{R}$. $\mathbf{x}(k) = [x_1(k), \dots, x_N(k)]^T$ is the joint estimate and $\mathbf{r}(k) = [r_1(k), \dots, r_N(k)]^T$ is the joint input of the network. Nodes, by means of $\mathbf{x}(k)$, cooperate to track the average $\bar{r}(k) = \frac{1}{N}\sum_{i=1}^N r_i(k)$ by exchanging information with their neighbors $j \in \mathcal{N}_i$.

The multi-stage discrete time dynamic average consensus algorithms presented by Franceschelli and Gasparri ([7], [8]) solve the problem for two cases: the synchronous and time-invariant topology, and the asynchronous and randomized topology. The former is solved by the following protocol:

$$\mathbf{x}^s(k+1) = (\mathbf{I} - \epsilon\mathbf{L})\mathbf{x}^s(k) + \alpha(\mathbf{x}^{s-1}(k) - \mathbf{x}^s(k)). \qquad (4)$$

Here, $s = \{1, \dots, m\}$ denotes the stages of the filter, $\mathbf{x}^s(k)$ is the estimate at instant k and stage s, $\mathbf{x}^0(k) = \mathbf{r}(k)$, parameter $\epsilon < \frac{1}{2D_{\max}}$ to ensure stability, and $\alpha \in (0, 1)$ is a parameter that trades-off steady-state accuracy and convergence speed. From (4) it is seen that the protocol is a chain of linear discrete dynamic average consensus filters indexed by s.

The asynchronous and randomized algorithm follows the same multi-stage architecture. In this case, each node selects randomly and at each instant a single node $j \in \mathcal{N}_i$ to communicate. For a given edge (i, j), the algorithm is:

$$\mathbf{x}^s(k+1) = \mathbf{P}_{ij}\mathbf{x}^s(k) + \frac{\alpha}{\mathbf{D}_{ii}}\mathbf{p}_i\mathbf{p}_i^T\mathbf{x}^{s-1}(k), \qquad (5)$$

with $\mathbf{P}_{ij} = \mathbf{I} + \frac{\mathbf{p}_i\mathbf{p}_j^T}{2} - \frac{(1+2/\mathbf{D}_{ii})\mathbf{p}_i\mathbf{p}_i^T}{2}$ and $\mathbf{p}_i \in \mathbb{R}^N$ a vector of zeros except for the $i-$th element, which is equal to 1.

As shown in [7], [8], the steady-state error and convergence rate of protocols (4) and (5) are tied together by α: for a fixed m, small values of α lead to quick convergence but large steady-state error and vice versa. The trade-off can be alleviated by m, but at the expense of an increase of computation, memory and convergence time at final stages.

[1]**Notation**: lower-case is for scalars, bold lower-case for vectors, bold capital for matrices, and calligraphic capital for sets. We use $\geq$ for greater than or equal, $\succeq$ for positive semidefiniteness, $\sigma(\ )$ for the eigenvalues of a matrix, ^{T} for the transpose, $\mathsf{car}(\ )$ for the cardinality of a set, $\mathsf{E}[\ ]$ for the expectancy operator, $|\ |$ the absolute value, and $||\ ||$ for the 2-norm of a vector/matrix. $\mathbf{P}_{ij}$ (resp. $\mathbf{p}_i$) denotes the ij-th (resp. i-th) element of matrix $\mathbf{P}$ (resp. vector $\mathbf{p}$). $\mathbf{I}$ is the identity matrix of appropriate dimensions, $\mathbf{0}$ is the zero matrix of appropriate dimension, and $\mathbf{1}$ is a column vector of ones.

Importantly, [8] proves that both algorithms are robust against arbitrary initializations and noisy and unbounded inputs due to the absence of integral control actions and input differences respectively. The next section proposes a second order diffusion protocol that, by adding an additional parameter, decouples the steady-state error and the convergence speed, overcoming the undesired trade-off in the multi-stage protocols.

As a remark, the stages of protocols (4), (5) run in parallel, in the sense that, to estimate at stage s and instant $k+1$, we only need information from $s-1$ and s at time k.

III. Proposed consensus algorithm

In this section, we present two algorithms. The first solves the problem of discrete time dynamic average consensus by means of a second order recurrence and the multi-stage architecture inspired by [7], [8]. The second solves the same problem but under asynchronous and randomized restrictions, where the nodes randomly communicate in a gossip-like style [20].

A. Accelerated Multi-Stage Filter

The first proposed consensus protocol is based on two steps. Given the current input $\mathbf{r}_i(k)$ and estimates from neighbors $\{x_j^s(k)\}_{s=0}^m \ \forall j \in \mathcal{N}_i \cup \{i\}$, node i updates $\{x_i^s(k)\}_{s=0}^m$ following the multi-stage filter in Eq. (4), obtaining $\{\tilde{x}_i^s(k)\}_{s=0}^m$, where $\tilde{x}_i^s(k) \in \mathbb{R}$ for $s = 1, \dots, m$ is a temporal estimate used in the second step of the protocol. The updated estimate is then corrected using the second order method in Eq. (3), leading to the next estimate $\{x_i^s(k+1)\}_{s=0}^m$. Algorithm 1 details the protocol.

Algorithm 1 Accelerated Multi-Stage Dynamic Consensus Protocol at node i

1: **State of agent**: $x_i^s(-1)$ and $x_i^s(0)$, for $s = 1, \dots, m$
2: **Parameters**: $\gamma \in (1,2)$, $\epsilon \in (0, \frac{1}{2D_{\max}})$, $\alpha \in (0, \frac{1}{\gamma})$
3: **while** True **do**
4: Measure $r_i(k)$
5: Gather $x_j^s(k)$ for $s = 1, \dots, m$ and $j \in \mathcal{N}_i$
6: Update $x_i^s(k)$ for $s = 1, \dots, m$ as follows:

$$\tilde{x}_i^s(k) = x_i^s(k) - \sum_{j \in \mathcal{N}_i} \epsilon (x_i^s(k) - x_j^s(k)) + \alpha (x_i^{s-1}(k) - x_i^s(k))$$
$$x_i^s(k+1) = \gamma(\tilde{x}_i^s(k)) + (1-\gamma) x_i^s(k-1) \ \forall s$$

7: **end while**

Note that Algorithm 1 is equal to (4) for $\gamma = 1$. First, for space convenience, in the following we use the sub-index $_k$ to abbreviate (k). We now prove the convergence properties of the proposed protocol. The next result shows that the second order method does not alter the steady-state properties of the original multi-stage filter in [7], [8], which is important in the subsequent results about the convergence rate of the accelerated protocol.

Proposition 1: Assume that $\mathbf{r}_k = \mathbf{r}$ is constant, $\mathcal{G}$ connected, $\alpha \in (0,1)$, and $\epsilon \in (0, \frac{1}{2D_{\max}})$. Then, the steady-state equilibrium $\mathbf{x}^{m,*}$ of the protocol in (4) and Algorithm 1 is

$$\mathbf{x}^{m,*} = \bar{r}\mathbf{1} + \sum_{i=2}^{N} \left(\frac{1}{1 + \epsilon \lambda_2(\mathbf{L})/\alpha} \right)^m \mathbf{v}_{r,i}(\mathbf{L}) \mathbf{v}_{l,i}^T(\mathbf{L}) \mathbf{r} \quad (6)$$

Proof: From Algorithm 1 and using the matrix form, the operations at stage $s = 1$ and steady-state can be written in a single equation

$$\mathbf{x}^{1,*} = \gamma(\mathbf{I} - \epsilon\mathbf{L})\mathbf{x}^{1,*} + \gamma\alpha\mathbf{r} - \gamma\alpha\mathbf{x}^{1,*} + (1-\gamma)\mathbf{x}^{1,*}. \quad (7)$$

This leads to $\left((1 + \gamma\alpha - 1 + \gamma - \gamma)\mathbf{I} + \gamma\epsilon\mathbf{L}\right)\mathbf{x}^{1,*} = \gamma\alpha\mathbf{r}$. Dividing both sides by γ, and concatenating the m stages,

$$\mathbf{x}^{m,*} = (\alpha\mathbf{I} + \epsilon\mathbf{L})^{-1}\alpha^m\mathbf{r}, \quad (8)$$

which is the steady-state equilibrium in the proof of Theorem 3.1 in [8]. The rest of the proof follows from there. ■

Corollary 1: Assume that $\mathbf{r}_k = \mathbf{r}$ is constant and $\mathcal{G}$ connected. Then, the 2-norm of the error at equilibrium of a network under Algorithm 1 is

$$||\bar{r}\mathbf{1} - \mathbf{x}^{m,*}|| \leq (N-1) \left(\frac{1}{1 + \epsilon\lambda_2(\mathbf{L})/\alpha} \right)^m ||\bar{r}\mathbf{1} - \mathbf{r}||. \quad (9)$$

The statement is a direct consequence of Proposition 1 and Theorem 3.2 in [7], showing that the steady-state error decreases when: the graph is more connected (greater $\lambda_2(\mathbf{L})$), the protocol has more stages (greater m), and α gets smaller. The following result proves necessary equivalences to demonstrate the convergence properties of Algorithm 1.

Proposition 2: Let $\mathbf{y}_k^s = \mathbf{x}_k^s - \mathbf{x}^{s,*}$ be the error at the s-th stage of the filter in Algorithm 1. Then, the error dynamics can be expressed as

$$\begin{pmatrix} \mathbf{y}_{k+1}^s \\ \mathbf{y}_k^s \end{pmatrix} = \hat{\mathbf{Q}} \begin{pmatrix} \mathbf{y}_k^s \\ \mathbf{y}_{k-1}^s \end{pmatrix} + \alpha\hat{\mathbf{R}} \begin{pmatrix} \Delta\mathbf{u}_k^s \\ \Delta\mathbf{u}_{k-1}^s \end{pmatrix}, \quad (10)$$

where $\hat{\mathbf{Q}} = \begin{pmatrix} \gamma\mathbf{Q} & (1-\gamma)\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}$, $\hat{\mathbf{R}} = \begin{pmatrix} \mathbf{R} & \mathbf{R}(1-\gamma) \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$, $\mathbf{Q} = (1-\alpha)\mathbf{I} - \epsilon\mathbf{L}$, $\mathbf{R} = (\alpha\mathbf{I} + \epsilon\mathbf{L})^{-1}$, $\Delta\mathbf{u}_k^s = \mathbf{u}_k^s - \mathbf{u}_{k+1}^s$, $\mathbf{u}_k^1 = \mathbf{r}_k$ and $\mathbf{u}_k^s = \mathbf{x}_k^{s-1}$ for $s = 2, \dots, m$.

Proof: Using the steps in Algorithm 1 in a single operation and matrix form,

$$\begin{aligned} \mathbf{y}_{k+1}^s = &\gamma\mathbf{Q}\mathbf{y}_k^s + \gamma\mathbf{x}^{s,*} - \mathbf{x}^{s,*} + (1-\gamma)\mathbf{x}_{k-1}^s = \\ &\gamma\mathbf{Q}\mathbf{y}_k^s + \gamma\mathbf{x}^{s,*} - \mathbf{x}^{s,*} + (1-\gamma)\mathbf{y}_{k-1}^s + (1-\gamma)\mathbf{x}^{s,*} \end{aligned} \quad (11)$$

Eq. (11) can be rewritten using Eq. (8):

$$\mathbf{y}_{k+1}^s = \gamma\mathbf{Q}\mathbf{y}_k^s + (1-\gamma)\mathbf{y}_{k-1}^s + \alpha\mathbf{R}\Delta\mathbf{u}_k^s + (1-\gamma)\alpha\mathbf{R}\Delta\mathbf{u}_{k-1}^s. \quad (12)$$

Eq. (10) follows from Eq.(12), concluding the proof. ■

Now we are ready to prove the convergence rate of Algorithm 1 .

Theorem 1: Consider a network that executes Algorithm 1 with $\mathbf{r}_k = \mathbf{r}$ constant, with $\alpha \in (0, \frac{1}{\gamma})$, $\gamma \in (1,2)$ and $\epsilon \in (0, \frac{1}{2D_{\max}})$. Then, the convergence rate for the s-th stage is $\beta^s = 1 - \alpha\gamma$.

Proof: Let us consider the following Lyapunov function:

$$V_k^s = \begin{pmatrix} \mathbf{y}_k^{s,T} & \mathbf{y}_{k-1}^{s,T} \end{pmatrix} \begin{pmatrix} \mathbf{y}_k^{s,T} & \mathbf{y}_{k-1}^{s,T} \end{pmatrix}^T = \hat{\mathbf{y}}_k^s \hat{\mathbf{y}}_k^{s,T}, \quad (13)$$

with $\hat{\mathbf{y}}_k^s = \begin{pmatrix} \mathbf{y}_k^{s,T} & \mathbf{y}_{k-1}^{s,T} \end{pmatrix}$. Then, the Lyapunov difference is

$$\Delta V_k^s = V_{k+1}^s - V_k^s = \hat{\mathbf{y}}_{k+1}^s \hat{\mathbf{y}}_{k+1}^{s,T} - \hat{\mathbf{y}}_k^s \hat{\mathbf{y}}_k^{s,T}. \quad (14)$$

Exploiting Eq. (10) and the fact that $\mathbf{r}_k$ is constant, Eq. (14) leads to

$$\Delta V_k^s = \hat{\mathbf{y}}_k^s (\hat{\mathbf{Q}}\hat{\mathbf{Q}}^T - \mathbf{I}) \hat{\mathbf{y}}_k^{s,T}. \quad (15)$$

Now, since $\gamma \in (1,2)$ and the spectral radius of $\mathbf{Q}$ is less than 1 by construction, it can be shown that Eq. (15) is upper-bounded by the following inequality:

$$\Delta V_k^s \leq -\hat{\mathbf{y}}_k^s \begin{pmatrix} \gamma^2\mathbf{I} - \gamma^2\mathbf{Q}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \hat{\mathbf{y}}_k^{s,T} = -\hat{\mathbf{y}}_k^s \mathbf{F} \hat{\mathbf{y}}_k^{s,T}. \quad (16)$$

The eigenvalues of matrix $\mathbf{F}$ are the eigenvalues of $\gamma^2\mathbf{I} - \gamma^2\mathbf{Q}^2$ and $\mathbf{I}$. If $\alpha < \frac{1}{\gamma}$, then $\gamma^2\mathbf{I} - \gamma^2\mathbf{Q}^2 \succeq \gamma\mathbf{I} - \gamma\mathbf{Q}$ and $\sigma(\gamma\mathbf{I} - \gamma\mathbf{Q}) > 0$. Therefore, $\mathbf{F}$ is positive definite, and

$$V_{k+1}^s - V_k^s \leq -\bar{\lambda}\hat{\mathbf{y}}_k^s \hat{\mathbf{y}}_k^{s,T} = -\bar{\lambda} V_k^s. \quad (17)$$

It turns out that $\bar{\lambda} = \min_{\lambda_i \in \sigma(\gamma\mathbf{I}-\gamma\mathbf{Q}) \cup \sigma(\mathbf{I})}\{\lambda_i\} = \alpha\gamma$. Thus, $V_{k+1}^s \leq (1-\alpha\gamma)V_k^s$. ■

Once the convergence rate of Algorithm 1 is proved, we can compare it with its non-accelerated counterpart.

Proposition 3: Algorithm 1 always converges faster than the protocol in Eq. (4) if $\gamma \in (1,2)$.

Proof: The convergence rate of the protocol in Eq. (4) is $\beta^s = 1-\alpha$, which coincides with the convergence rate obtained by Franceschelli and Gasparri ([7]). By comparing both convergence rates, we have that $1-\alpha > 1-\alpha\gamma$ for $\gamma \in (1,2)$. Thus, the bound in Eq. (17) is smaller, and the decrease in the Lyapunov function defined in Eq. (13) is greater, which implies a faster convergence of Algorithm 1 than the protocol in Eq. (4). ■

These results draw interesting properties. First, for any choice of m and α of the original algorithm, such that α also fulfills the condition $\alpha \in (0, 1/\gamma)$ with $\gamma \in (1,2)$, then Algorithm 1 always converges faster than the non-accelerated protocol (4). Besides, while α ties together steady-state and convergence rate in both the original and accelerated algorithm, on the original protocol we can only exploit m to further tune the steady-state error. Meanwhile, on Algorithm 1, by introducing γ, we have an additional degree of freedom to also further tune the convergence, independent from the steady-state error. Thus, the accelerated protocol provides more flexibility in simultaneously tuning the steady-state error and convergence rate.

All the previous results address the case where the input is constant. The following result considers a dynamic input.

Proposition 4: Consider a time-varying input $\mathbf{r}_k$ and a network under Algorithm 1, with $\alpha \in (0, \frac{1}{\gamma})$. Besides, define $\Delta\mathbf{u}_\infty^s = \sup_{k=0,\ldots,\infty} \Delta\mathbf{u}_k^s$. Then, the next ISS property holds:

$$||\mathbf{y}_k^s|| \leq (1-\gamma\alpha)^k ||\mathbf{y}_0^s|| + \frac{1}{\alpha\gamma}\Delta\mathbf{u}_\infty^s. \quad (18)$$

Proof: The result follows from Input-to-State Stability results for linear systems. From Eq. (12) we have that

$$||\mathbf{y}_k^s|| \leq ||\mathbf{K}^k|| \cdot ||\mathbf{y}_0^s|| + \alpha \left(\sum_{h=0}^{k} ||\mathbf{K}^h|| ||\mathbf{R}|| \right) \Delta\mathbf{u}_\infty^s, \quad (19)$$

with $\mathbf{K} = (\gamma\mathbf{Q} + (1-\gamma)\mathbf{I})$. First, note that $||\mathbf{R}|| \leq \frac{1}{\alpha}$. Second, $||\mathbf{K}|| \leq (1-\gamma) + \gamma(1-\alpha) = 1-\gamma\alpha$. Then,

$$||\mathbf{y}_k^s|| \leq (1-\gamma\alpha)^k ||\mathbf{y}_0^s|| + \alpha \left(\sum_{h=0}^{k} \frac{(1-\gamma\alpha)^h}{\alpha} \right) \Delta\mathbf{u}_\infty^s. \quad (20)$$

Eq. (20) directly yields to Eq. (18). ■

Therefore, under dynamic inputs the system still converges. Moreover, the convergence rate only depends on the variations of the signal.

Corollary 2: The convergence rate of Algorithm 1 is faster than the convergence rate of the original filter in (4) under time-varying input signals $\mathbf{r}_k$, $\alpha \in (0, \frac{1}{\gamma})$ and $\gamma \in (1,2)$.

Proof: In Proposition 3.5 from ([7]) it is shown that

$$||\mathbf{y}_k^s|| \leq (1-\alpha)^k ||\mathbf{y}_0^s|| + \frac{1}{\alpha}\Delta\mathbf{u}_\infty^s. \quad (21)$$

Comparing Eqs. (20) and (21), the first term experiences a faster decay in (20) since $1-\gamma\alpha < 1-\alpha$. The second term is lower in (20) since $\frac{1}{\alpha\gamma} < \frac{1}{\alpha}$. ■

In summary, the proposed algorithm proves to enhance the convergence speed towards the dynamic average consensus without modifying the steady-state error at any of the stages of the filter. This is achieved by using an additional memory slot for the estimate at the previous instant of time.

To design the algorithm, a possible way of proceed is the following: (i) set $\beta^s \in (0,1)$, which implies that $\alpha\gamma = 1-\beta^s \in (0,1)$; (ii) therefore, $\alpha = (1-\beta^s)/\gamma$, which is always feasible because $(1-\beta^s)/\gamma < 1/\gamma$; (iii) at this point, set the desired steady-state error and choose m according to Corollary 1 to achieve the desired performance. Notably, the computational and memory cost of Algorithm 1 grows linearly with the number of stages, and thus increasing m is a scalable design decision, considering also the current advances in PMEMS technology. For instance, if we consider the Ethernet protocol, with a frame between 64 and 1518 bytes and a 18-byte header, we can send between 11 and 375 floating point numbers, equivalent to $m \in [11, 375]$. Thus, we can fully exploit the real structure of communication networks. In addition, by leveraging the results provided in [8], Algorithm 1 inherits the robustness against initialization and noisy unbounded inputs.

B. Accelerated Asynchronous Randomized Protocol

The second contribution is the asynchronous and randomized version of Algorithm 1. Now, instead of using all the estimates from the neighborhood, node i selects one of its neighbors $j \in \mathcal{N}_i$ according to an independent and identical distribution (i.i.d.) with uniform probability. The rest of the protocol follows the same reasoning, detailed in Algorithm 2.

The main result in this section shows the equivalence in expectation between Algorithm 2 and Algorithm 1. Then, the results provided in subsection III-A can be extrapolated to the asynchronous and randomized setting.

Theorem 2: Consider a network under Algorithm 2 with $\mathcal{G}$ connected, $\mathbf{r}(k) = \mathbf{r}$ constant, $\alpha \in (0, \frac{1}{\gamma})$, and $\gamma \in (1,2)$. If the sequence of selected edges is i.i.d. with uniform probability, then Algorithm 2 preserves the steady-state error properties of the original filter in (5), i.e,:

- $\mathbf{x}^m(k)$ converges in distribution to a random variable $\mathbf{x}_\infty^m$, and this distribution is unique.
- $\lim_{k\to\infty} \mathsf{E}[\mathbf{x}^m(k)] = \mathsf{E}[\mathbf{x}_\infty^m] = \mathbf{x}^{m,*}$.

Proof: The first statement of the proof is a direct extrapolation of the proof of Theorem 6 in [8]. Algorithm 2 fulfills all the following requirements: discrete time, Schur

stability, affine dynamics and the sequence of edges is i.i.d. with uniform probability. Regarding the second statement, Theorem 4.1 in [7] proves

$$\mathsf{E}[\bar{\mathbf{x}}^1(k+1)] = (\mathbf{I} - \epsilon'\mathbf{L})\mathsf{E}[\mathbf{x}^1(k)] + \alpha'(\mathbf{r} - \mathsf{E}[\mathbf{x}^1(k)]), \quad (22)$$

with $\epsilon' = \frac{1}{2\mathsf{car}(\mathcal{E})}$ and $\alpha' = \frac{\alpha}{\mathsf{car}(\mathcal{E})}$. Accordingly, the protocol in Algorithm 2 is

$$\begin{aligned}\mathsf{E}[\mathbf{x}^1(k+1)] =& \gamma(\mathbf{I} - \epsilon'\mathbf{L})\mathsf{E}[\mathbf{x}^1(k)] + \gamma\alpha'(\mathbf{r} - \mathsf{E}[\mathbf{x}^1(k)]) \\ &+ (1-\gamma)\mathsf{E}[\mathbf{x}^1(k-1)]).\end{aligned} \quad (23)$$

The rest of the proof follows from the fact that, if we replace $\mathsf{E}[\mathbf{x}^1(k)] = \mathsf{E}[\mathbf{x}^1(k+1)] = \mathsf{E}[\mathbf{x}^1(k-1)] = \mathsf{E}[\mathbf{x}^{m,*}(k)]$, then Eq. (23) and Eq. (7) are the same, so the same procedure can be followed to prove the second statement. ■

Algorithm 2 Accelerated Asynchronous Randomized Multi-Stage Dynamic Consensus Protocol at node i

1: **State of agent**: $x_i^s(-1)$ and $x_i^s(0)$, for $s = 1, \dots, m$
2: **Parameters**: $\gamma \in (1,2)$, $\alpha \in (0, \frac{1}{\gamma})$, $D_i = \mathbf{L}_{ii}$
3: **while** True **do**
4: Measure $r_i(k)$
5: Select a random neighbor and gather $x_j^s(k)$
6: Update $x_i^s(k)$ for $s = 1, \dots, m$ as follows:
$\tilde{x}_i^s(k) = (x_i^s(k) + x_j^s(k))/2 + \alpha/D_i(x_i^{s-1}(k) - x_i^s(k))$
$x_i^s(k+1) = \gamma(\tilde{x}_i^s(k)) + (1-\gamma)x_i^s(k-1)\ \forall s$
7: **end while**

Therefore, Algorithms 1 and 2 are equivalent in expectation. Interestingly, another advantage of our accelerated proposal is that α can be tuned to have lower values while preserving the convergence speed, leading to lower noise. In addition, Algorithm 2 is easier to implement in a real device because no synchronization is needed, and an inherent robustness against delays and packet losses is achieved, following the considerations in [8]. Besides, since the tracking properties are independent on the initialization, the protocol is robust against a varying number of nodes. Regarding the estimates' variance, the sequence of consensus stages act as a filter, reducing the variance from stage to stage. This is observed in Algorithm 2, where the neighboring estimates are averaged and corrected by the estimate from the previous stage. We empirically verify this fact in section IV, leaving its theoretical characterization for future work.

Finally, while protocols (4) and (5) allow $\alpha \in (0,1)$, Algorithms 1 and 2 constrain $\alpha \in (0, 1/\gamma)$. This is not a limitation of the proposed algorithms. The discussion after Corollary 2 shows how to ensure that $\alpha < 1/\gamma$ always.

IV. Illustrative Examples

To evaluate Algorithms 1 and 2, we consider an undirected graph of $N = 8$ nodes with $\mathcal{E}$={(1,2),(1,5),(1,8),(2,3),(2,8),(3,4),(3,6),(4,8),(6,7),(7,8)} The parameters are $\alpha = 0.04$, $\epsilon = 0.01$, and $m = 5$. The signals $r_i(k)$ evolve according to a uniform random process, such that, every 2000 steps, $r_i(k) \sim \mathcal{U}(0,1)$ $\forall i \in \mathcal{V}$. Besides, we set the initial estimates for all stages as $\mathbf{x}^s(0)$=$[0.99, 0.27, 0.02, 0.48, 0.18, 0.24, 0.65, 0.50]^T$. This

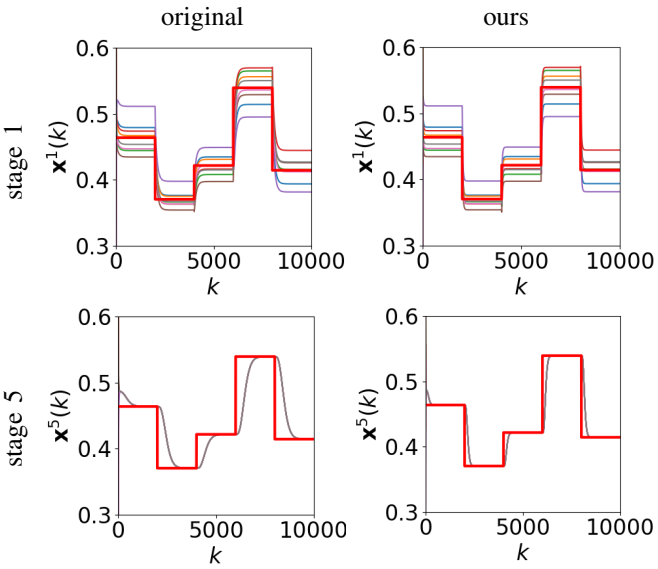

Fig. 1. Simulation results for (left) the original multi-stage and (right) the accelerated multi-stage algorithm (ours). The red bold line is the average to be tracked, whereas the local estimates are depicted in thin random color lines.

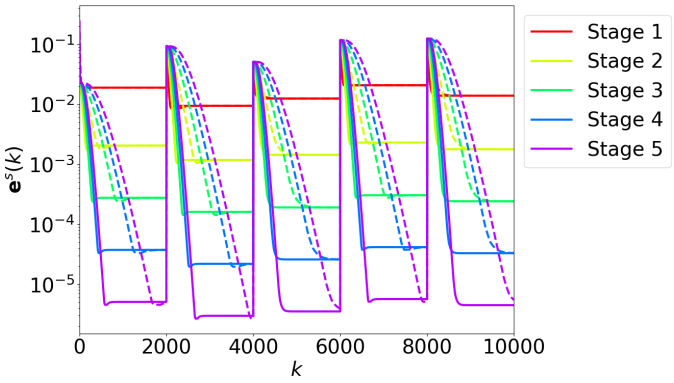

Fig. 2. Evolution of the absolute error between the average estimate across the network and the average of the input signals. The accelerated multi-stage filter is in solid lines, whereas the original multi-stage filter is in dashed lines.

initialization is random, according to the inherited robustness of the multi-stage protocol [8]. The value of γ can be computed by means of distributed algorithms that estimate the algebraic connectivity of the graph (see, e.g., [25], [26]).

The evolution of the estimates for the synchronous and non-randomized algorithms is shown in Fig. 1. With the additional memory slot, the convergence time has been substantially improved by Algorithm 1 (Fig. 1 (right)) compared to the original non-accelerated protocol (Fig. 1 (left)), while maintaining the steady-state performance. To better compare the steady-state error and convergence speed, Fig. 2 draws the absolute error between the average estimate across the network $e^s(k) = \frac{1}{N}\sum_{i=1}^{N}|x_i^s(k) - r_i(k)|$ and the average of the input signals $\mathbf{e}(k) = [e^1(k), \dots, e^m(k)]^T$. For the same convergence speed, the accelerated filter can be designed with more stages and improve the steady-state error in various orders of magnitude.

In the randomized algorithms, we set $\alpha = 0.0005$. Besides,

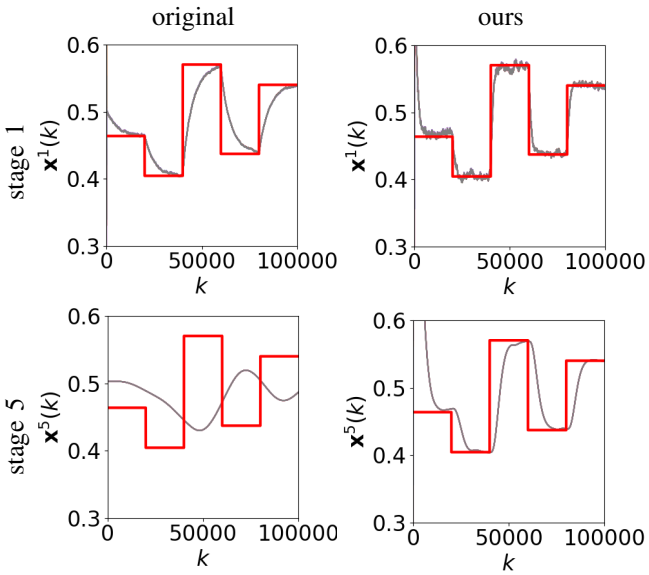

Fig. 3. Simulation results for (left) the randomized original multi-stage and (right) the accelerated randomized multi-stage algorithm (ours). The style is the same in Fig. 1.

since the topology changes arbitrarily, we fix $\gamma = 1.7$. Finally, the signals $r_i(k)$ change every 20000 steps. Fig. 3 represents the result of the experiments. With a low value of α, the noise due to the randomized links is filtered. The accelerated filter can compensate the degradation in convergence speed, while the original filter is too slow to converge before the signals change. Thus, a single additional memory slot per stage leads to an acceleration that overcomes the trade-off between speed and accuracy present in the original protocols in [7], [8].

V. Conclusions

This paper has presented two novel accelerated discrete time dynamic average consensus protocols based on a sequence of proportional consensus filters and a second order recurrence. The combination overcomes the trade-off between convergence speed and steady-state error, while achieving robustness against initialization and input noise. The multi-stage scheme can arbitrarily reduce the steady-state error, but this implies a slow convergence. Thanks to the second order recurrence, the convergence is sped up, counteracting this drawback, specially at final stages of the protocol. These conclusions are shared by the asynchronous and randomized version of the algorithm. In the latter, parameter α manifests a trade-off between convergence speed, noise and average steady-state error that the acceleration due to the second order method compensates, while achieving robustness against communication delays and packet losses.

References

[1] F. Garin and L. Schenato, "A survey on distributed estimation and control applications using linear consensus algorithms," in *Networked control systems*. Springer, 2010, pp. 75–107.

[2] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on dynamic average consensus: The problem, its applications, and the algorithms," *IEEE Control Systems Magazine*, vol. 39, no. 3, pp. 40–72, 2019.

[3] M. Franceschelli, A. Pilloni, and A. Gasparri, "Multi-agent coordination of thermostatically controlled loads by smart power sockets for electric demand side management," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 731–743, 2020.

[4] E. Sebastián, E. Montijano, and C. Sagüés, "All-in-one: Certifiable optimal distributed Kalman filter under unknown correlations," in *IEEE Conference on Decision and Control*, 2021, pp. 6578–6583.

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

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

[7] M. Franceschelli and A. Gasparri, "Multi-stage discrete time dynamic average consensus," in *IEEE Conference on Decision and Control*, 2016, pp. 897–903.

[8] ——, "Multi-stage discrete time and randomized dynamic average consensus," *Automatica*, vol. 99, pp. 69–81, 2019.

[9] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *IEEE Conference on Decision and Control*. IEEE, 2005, pp. 6698–6703.

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

[11] S. Nosrati, M. Shafiee, and M. B. Menhaj, "Dynamic average consensus via nonlinear protocols," *Automatica*, vol. 48, no. 9, pp. 2262–2270, 2012.

[12] J. George and R. A. Freeman, "Robust dynamic average consensus algorithms," *IEEE Transactions on Automatic Control*, vol. 64, no. 11, pp. 4615–4622, 2019.

[13] B. Ghosh, S. Muthukrishnan, and M. H. Schultz, "First and second order diffusive methods for rapid, coarse, distributed load balancing," in *ACM symposium on Parallel algorithms and architectures*, 1996, pp. 72–81.

[14] J. Liu and A. S. Morse, "Accelerated linear iterations for distributed averaging," *Annual Reviews in Control*, vol. 35, no. 2, pp. 160–165, 2011.

[15] 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, 2008.

[16] E. Montijano, J. I. Montijano, and C. Sagues, "Chebyshev polynomials in distributed consensus applications," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 693–706, 2012.

[17] B. Van Scoy, R. A. Freeman, and K. M. Lynch, "A fast robust nonlinear dynamic average consensus estimator in discrete time," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 191–196, 2015.

[18] V. S. Donge, B. Lian, F. L. Lewis, and A. Davoudi, "Accelerated reinforcement learning via dynamic mode decomposition," *IEEE Transactions on Control of Network Systems*, 2023.

[19] T. Sadamoto, A. Chakrabortty, and J.-i. Imura, "Fast online reinforcement learning control using state-space dimensionality reduction," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 1, pp. 342–353, 2020.

[20] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[21] J. Liu, S. Mou, A. S. Morse, B. D. Anderson, and C. Yu, "Deterministic gossiping," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1505–1524, 2011.

[22] F. Fagnani and S. Zampieri, "Average consensus with packet drop communication," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 102–133, 2009.

[23] H. Moradian and S. S. Kia, "On robustness analysis of a dynamic average consensus algorithm to communication delay," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 633–641, 2018.

[24] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009, vol. 27.

[25] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of Laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013.

[26] E. Montijano, J. I. Montijano, and C. Sagues, "Fast distributed algebraic connectivity estimation in large scale networks," *Journal of the Franklin Institute*, vol. 354, no. 13, pp. 5421–5442, 2017.