Autonomous Robots (2019) 43:1989–2010
https://doi.org/10.1007/s10514-019-09845-4

Distributed iterative learning control for multi-agent systems

Theoretic developments and application to formation flying

Andreas Hock[1] · **Angela P. Schoellig**[1]

Received: 17 August 2016 / Accepted: 15 February 2019 / Published online: 27 March 2019

Abstract
The goal of this work is to enable a team of quadrotors to learn how to accurately track a desired trajectory while holding a given formation. We solve this problem in a distributed manner, where each vehicle has only access to the information of its neighbors. The desired trajectory is only available to one (or few) vehicle(s). We present a distributed iterative learning control (ILC) approach where each vehicle learns from the experience of its own and its neighbors' previous task repetitions and adapts its feedforward input to improve performance. Existing algorithms are extended in theory to make them more applicable to real-world experiments. In particular, we prove convergence of the learning scheme for any linear, causal learning function with gains chosen according to a simple scalar condition. Previous proofs were restricted to a specific learning function, which only depends on the tracking error derivative (D-type ILC). This extension provides more degrees of freedom in the ILC design and, as a result, better performance can be achieved. We also show that stability is not affected by a linear dynamic coupling between neighbors. This allows the use of an additional consensus feedback controller to compensate for non-repetitive disturbances. Possible robustness extensions for the ILC algorithm are discussed, the so-called Q-filter and a Kalman filter for disturbance estimation. Finally, this is the first work to show distributed ILC in experiment. With a team of two quadrotors, the practical applicability of the proposed distributed multi-agent ILC approach is attested and the benefits of the theoretic extension are analyzed. In a second experimental setup with a team of four quadrotors, we evaluate the impact of different communication graph structures on the learning performance. The results indicate, that there is a trade-off between fast learning convergence and formation synchronicity, especially during the first iterations.

Keywords Iterative learning control · Multi-agent systems · Distributed control · Quadrotor control

1 Introduction and related work

Multi-agent systems (MAS) and machine learning are two exciting trends in robotics. In the past decades, theoretic contributions to MAS have come from fields such as biology, computer science, and control theory. One problem of particular interest is *consensus*, which is concerned with all agents agreeing on some quantity of interest by only communicating with their neighbors. Many other problems can be transformed into a consensus problem; examples include flocking, rendezvous or formation control [1]. Distributed algorithms have been developed that achieve consensus among autonomous agents without a central control unit, see for example [2–4]. Machine learning, on the other hand, aims to make autonomous systems more capable by enabling them to adapt to unknown situations, automatically correct for modeling errors, and improve their performance without human intervention. As the number of autonomous systems increases in all areas, giving rise to industrial and service robots, commercial drones, or self-driving cars, for example, the question arises how their cooperation can be improved

This research was supported in part by NSERC Grant RGPIN-2014-04634, the Connaught New Researcher Award and the Baden-Württemberg-STIPENDIUM.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10514-019-09845-4) contains supplementary material, which is available to authorized users.

✉ Andreas Hock
andreas.hock@robotics.utias.utoronto.ca

Angela P. Schoellig
schoellig@utias.utoronto.ca

[1] Dynamic Systems Lab (DSL), University of Toronto Institute for Aerospace Studies (UTIAS), Toronto, Canada

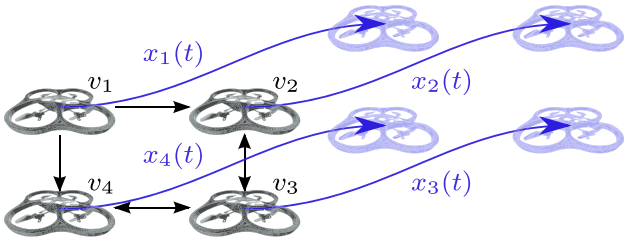

Fig. 1 A team of quadrotors learns to accurately track a desired trajectory while holding a given formation. The black arrows between the vehicles v_i represent the communication paths. The blue arrows depict the vehicle trajectories $x_i(t)$ (Color figure online)

and, therefore, how MAS and machine learning can be combined.

In this work, we focus on iterative learning control (ILC) approaches, where a system learns to track a desired trajectory by repetitively executing the same task. Based on the tracking error from previous trials, the feedforward input is adapted to gradually improve performance. As such, ILC is able to compensate for repetitive disturbances. Initially developed in 1984 [5], it has since been studied widely in theory and experiments, see [6] for an introduction and overview. In [7], ILC was used for the first time to improve the trajectory tracking of a single quadrotor vehicle. Another application to quadrotors can be found in [8], where high performance, periodic maneuvers are learned. Whereas they use more sophisticated input-update rules than we propose in this work, both these papers only consider one vehicle. A first experimental approach to couple two individual ILC systems is demonstrated in [9]. Their so-called cross-coupled ILC enhances the precision motion control of a two-axis robotic testbed.

Distributed ILC achieves formation control of MAS, where only a subset of agents has direct access to the reference trajectory and only neighboring agents can communicate, see Fig. 1. The goal is to follow the reference with all agents holding a predefined formation. To achieve this, every agent updates its input trajectory based on the information about its own and its neighbors' trajectories during the last trial. The idea of distributed ILC was first introduced in 2009 [10], where stability was proven for a D-type input update rule; that is, the input for the next iteration is computed based on the previous input and the derivative of the tracking error. Furthermore, this first paper assumes communication graphs in the form of directed spanning trees. The heterogeneous agent dynamics are described in continuous time and are assumed to be nonlinear with relative degree one. In [11], the proof was extended to arbitrary directed graphs and the stability criterion was simplified for the case of homogeneous agents. Even time-varying dynamics can be handled with the approach in [11]. To extend the algorithm to systems of higher relative-degree, higher derivatives (or, for discrete-time systems, appropriate time-shifts) were used in the input update rule, see [12]. This approach is, however, restricted to linear agent dynamics, but holds for weighted and directed communication graphs.

All these papers [10–12] are using the same D-type input update rule. The shortcoming of this control strategy is a poor learning behavior; for example, a constant offset in the initial condition cannot be compensated for as D-type ILC only adapts the input based on errors in the velocity.

In this paper, we prove that for linear agent dynamics any causal learning function with gains chosen according to a simple scalar condition is stable. This allows more options in the choice of the input-update rule (beyond the typical D-type update rule) and thus faster learning convergence can typically be achieved. Moreover, a constant error in the initial position can now be compensated for by incorporating position errors into the learning function.

Several other multi-agent ILC approaches and extensions are found in the literature, including ones for switching graph structures [13] and adaptive [14], robust [15,16] or optimal [17] update laws. However, all of these approaches have only been implemented in simulation and, to the authors' best knowledge, multi-agent ILC has not been tested on a real system before.

Since there has not been any practical implementation yet, the question arises, what multi-agent ILC can be useful for. In [18], trajectory-keeping in satellite formation flying is presented as one possible application. Beyond, it could be used for a load lifting and transportation task executed by a team of autonomous robots, for example quadrotors. If the mass and the inertial properties of the load are unknown, the vehicles could try to lift it as long as they operate in certain bounds. If they get out of these (safety) limits, they can stop the execution and try again with updated trajectories. The possibility of gradually extending the time horizon of the trajectory is an advantage of ILC as shown in [7]. Other applications may include platooning, the formation driving of autonomous cars, especially trucks.

This paper demonstrates the proposed generalized multi-agent ILC algorithm in experiment on quadrotor vehicles. To compensate for non-repetitive disturbances occurring in real-world experiments, we include a distributed feedback controller, which improves the performance during single iterations. Furthermore, we show that neither the distributed feedback controller nor any other linear dynamic coupling between neighboring agents affect stability of the ILC algorithm. Additionally, robustness of the proposed ILC algorithm against non-repeating disturbances and noise, such as sensor or process noise, is increased by including a Q-filter and an optional Kalman-filter-based disturbance estimation. Extensive experiments were conducted to demonstrate the effectiveness of our algorithm and to highlight the effect of

the different extensions. Videos of the experiments with two and four vehicles, respectively, are found at:
https://youtu.be/Qw598DRw6-Q and
https://youtu.be/JppRu26eZgI.

This work is structured as follows: first, we introduce some definitions and basic terminology from graph theory. In Sect. 3, the distributed ILC algorithm is developed in theory and the stability conditions for an arbitrary linear causal learning function are derived. Based on these results, we show that a dynamic coupling between neighboring agents does not influence stability and propose a consensus feedback controller and further extensions to increase robustness. The developed algorithm is demonstrated on a real multi-agent system. To show the effectiveness of the presented ILC and its extensions, experimental results for a team of two quadrotors flying a horizontal motion are shown in Sect. 5. In a second experimental setup, where four vehicles are performing a vertical motion, we analyze the influence of different communication graphs. Finally, conclusions are provided in Sect. 7.

This work has been accepted for publication at the 2016 IEEE conference on decision and control (CDC) [19]. This paper includes the following major extensions compared to the conference paper:

- The error definition, see Sect. 3.1.2, includes an additional term to compute the mean of the relative distances to the neighbors, which allows the statement of Theorem 3.
- In Sect. 3.4, we prove that the additional feedback controller guarantees consensus in time-domain for the closed-loop multi-agent system with double-integrator agent dynamics. Furthermore, convergence of all agents' tracking errors to zero can be shown for reference trajectories with constant velocity.
- The robustness extensions in Sect. 3.5 complement the learning algorithm for practical usage.
- In Sect. 5, additional experiments with two quadrotors are shown to provide evidence of the improvements compared to the previous D-type multi-agent ILC algorithm and to show the benefits of the robustness extension.
- The developed learning scheme is applied to a larger system consisting of four vehicles in a second experimental setup. This allows us to analyze the influence of different communication structures in Sect. 6.

2 Preliminaries on graph theory

To describe the information exchange between agents, graph theory is commonly used. The vehicles are the nodes of the graph and the edges represent the information flow between vehicles, see Fig. 1. In the following, we provide necessary definitions and notation and state some useful properties. For more details about graph theory, see [20].

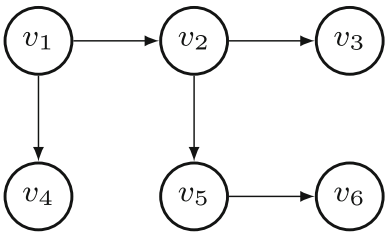

Fig. 2 Spanning tree structure for a directed graph. Node v_1 has no incoming edges and is called the root node. All other vertices have exactly one parent (one incoming edge) and can be connected to the root through directed paths

2.1 Definitions

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ denote a directed graph with a set of vertices $\mathcal{V}(\mathcal{G}) = \{v_i : i \in \{1, 2, \ldots, N\}\}$, and the edge set $\mathcal{E}(\mathcal{G}) \subseteq \{(v_i, v_j) : v_i, v_j \in \mathcal{V}(\mathcal{G})\}$. The edge (v_i, v_j) means that agent v_i receives information from v_j, in which case v_i is called the child and v_j the parent vertex. We define $\mathcal{A} = (a_{ij}) \in \mathbb{R}^{N\times N}$ as the adjacency matrix of $\mathcal{G}$ with elements representing the information exchange between any two agents; that is, $a_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}(\mathcal{G})$ and $a_{ij} = 0$ otherwise. The in-degree of node v_i is defined as $d_i^{\text{in}} = \sum_{j=1}^{N} a_{ij}$; that is, it describes the number of edges entering a node. The (in-degree) Laplacian matrix is defined as $\mathcal{L}_\mathcal{G} = \mathcal{D} - \mathcal{A}$, where $\mathcal{D} = diag(d_1^{\text{in}}, d_2^{\text{in}}, \ldots, d_N^{\text{in}})$. Dividing each non-zero row of the Laplacian by its in-degree d_i^{in} leads to the definition of the normalized Laplacian $\widetilde{\mathcal{L}}_\mathcal{G}$ according to [21].

A graph is called undirected if for every edge (v_i, v_j) the reverse edge (v_j, v_i) exists. In this case, the Laplacian matrix is symmetric.

If there exists a special vertex, the root, with no parents ($d^{\text{in}} = 0$), which all other nodes can be connected to through directed paths and every other vertex has exactly one parent, then the graph is called a spanning tree, see Fig. 2. Note that some authors also use the term 'rooted out-branching' to distinguish between directed and undirected graphs.

2.2 Laplacian spectrum

The concept of the Laplacian matrix is crucial for the theory of consensus and multi-agent control. For analyzing the stability of our learning algorithm, the eigenvalues of the Laplacian play an important role. In this section, we provide some relevant results.

Proposition 1 *The spectrum of the Laplacian matrices can be bounded using the Gershgorin circle theorem* [22]. *With* $\Re(\cdot)$

denoting the real part of a complex number, the eigenvalues $\lambda_i(\cdot), i \in \{1, \ldots, N\}$, *are bounded by:*

(i) *Laplacian* $\mathcal{L}_\mathcal{G}$*:*

$$0 \leq \Re(\lambda_i(\mathcal{L}_\mathcal{G})) \leq 2(N-1); \tag{1}$$

(ii) *Normalized Laplacian* $\widetilde{\mathcal{L}}_\mathcal{G}$*:*

$$0 \leq \Re(\lambda_i(\widetilde{\mathcal{L}}_\mathcal{G})) \leq 2. \tag{2}$$

Proof The proof follows directly from the Gershgorin circle theorem [22]. The diagonal entry and the sum of the absolute values of the non-diagonal entries in each row are equal and, therefore, zero is a lower bound of the spectrum. The upper bound for both, radius and center of the Gershgorin disk, is given by the maximum number of neighbors, which is $N-1$ in case (i). For the normalized Laplacian (ii), both values are equal to one. For further analysis of the Laplacian spectrum, see [21,23]. □

Remark 1 For undirected graphs the Laplacian is symmetric and, thus, all eigenvalues are real. For general graphs with more than two nodes, eigenvalues can appear in complex conjugate pairs.

From Proposition 1, we know that all eigenvalues lie in the right half plane of the complex plane or at 0. Additionally, they can be upper bounded. The following statement provides some information about the number of eigenvalues at the origin. A graph is said to contain a spanning tree if it can be constructed based on a spanning tree by adding additional edges. That is, some (improper) subset of its edges forms a spanning tree.

Proposition 2 *The (normalized) Laplacian always has one eigenvalue at 0. It has exactly one eigenvalue at zero if and only if the graph contains a spanning tree. If the graph is a spanning tree (no additional edges), all non-zero eigenvalues are equal to one* [20].

2.3 Reference trajectory as virtual leader

A discrete time reference trajectory $y_{des}(t)$ is given for all $t \in \{0, 1, \ldots, T\}$, $T < \infty$, which a certain subset of agents has access to.

We model the reference trajectory as an additional node v_0 and call it the virtual leader. The virtual leader has only out-going edges to those agents that have access to the reference y_{des}. The result is an extended graph $\mathcal{G}^*$. Let $b_i \in \{0, 1\}$ be defined for every agent analogously to the entries of the adjacency matrix: $b_i = 1$ if agent i has access to the reference trajectory, and $b_i = 0$ otherwise. The corresponding Laplacian $\mathcal{L}_{\mathcal{G}^*}$ is

$$\mathcal{L}_{\mathcal{G}^*} = \begin{bmatrix} 0 & 0_{1\times N} \\ -\mathrm{b} & \mathcal{L}_\mathcal{G} + \mathcal{B} \end{bmatrix} \tag{3}$$

with $\mathrm{b} = (b_1, b_2, \ldots, b_N)^T$, $\mathcal{B} = diag(\mathrm{b})$ and $0_{1\times N}$ denoting a matrix of size $(1 \times N)$ with all entries being zero. Normalizing $\mathcal{L}_{\mathcal{G}^*}$ leads to

$$\widetilde{\mathcal{L}}_{\mathcal{G}^*} = \begin{bmatrix} 0 & 0_{1\times N} \\ -W\mathrm{b} & W(\mathcal{L}_\mathcal{G} + \mathcal{B}) \end{bmatrix}, \tag{4}$$

where W is a diagonal weighting matrix,

$$W = diag\left(\frac{1}{d_1^{\text{in}} + b_1}, \frac{1}{d_2^{\text{in}} + b_2}, \ldots, \frac{1}{d_N^{\text{in}} + b_N}\right). \tag{5}$$

Note that d_i^{in} in (5) are the in-degrees of $\mathcal{L}_\mathcal{G}$. Without loss of generality, $(d_i^{\text{in}} + b_i)$ can be assumed to be non-zero, otherwise agent v_i would have neither the reference nor neighbors to learn from. To facilitate further derivations, we introduce

$$\widehat{\mathcal{L}}_{\mathcal{G}^*} = W(\mathcal{L}_\mathcal{G} + \mathcal{B}). \tag{6}$$

For readability, we skip the graph index $\mathcal{G}^*$ in the following and simply use the notation $\widehat{\mathcal{L}}$.

Proposition 3 *The eigenvalue spectrum of* $\widehat{\mathcal{L}}$ *can be bounded by*

$$0 \leq \Re(\lambda_i(\widehat{\mathcal{L}})) \leq 2. \tag{7}$$

Proof It can be seen in (4) that $\widetilde{\mathcal{L}}_{\mathcal{G}^*}$ is block triangular. Using the fact that the eigenvalues of a block triangular matrix are the combined eigenvalues of the diagonal blocks, the statement follows from Proposition 1. □

Proposition 4 *The matrix* $\widehat{\mathcal{L}}$ *has full rank and all its eigenvalues lie in the right half plane if and only if* $\mathcal{G}^*$ *contains a spanning tree.*

Proof If the extended graph $\mathcal{G}^*$ contains a spanning tree, node v_0 representing the virtual leader must be the root as it has in-degree $d_0^{\text{in}} = 0$. From Proposition 2 follows that $\mathcal{L}_{\mathcal{G}^*}$ and $\widetilde{\mathcal{L}}_{\mathcal{G}^*}$, respectively, only have one zero eigenvalue. With (4) and (6), we see that $\widehat{\mathcal{L}}$ is full rank. The statement then follows from Proposition 3. The necessary condition is evident, as the Laplacian of a graph not containing a spanning tree has more than one zero eigenvalue and, thus, $\widehat{\mathcal{L}}$ is not full rank. □

3 Developing the distributed learning algorithm

3.1 Problem formulation

Consider a system of N autonomous agents with index $i = \{1, 2, \ldots, N\}$. The communication between the agents can be described by some graph $\mathcal{G}$ and is restricted to position information. A certain subset of agents has access to the desired trajectory $y_{des} \in \mathbb{R}$ as described above. The goal is to track this reference signal with all agents simultaneously. The result can be directly applied to formation control by simply defining fixed or time-varying relative distances between the agents.

3.1.1 Agent dynamics

The linear discrete-time single-input, single-output dynamics of the ith agent during the k^{th} iteration of the learning, $k \in \{1, 2, \ldots\}$, and for sampling times $t \in \{0, 1, \ldots, T\}$ are described by

$$\begin{aligned} x_{i,k}(t+1) &= Ax_{i,k}(t) + Bu_{i,k}(t) \\ y_{i,k}(t) &= Cx_{i,k}(t), \end{aligned} \tag{8}$$

where $x_{i,k}(t) \in \mathbb{R}^n$ is the state vector, $u_{i,k}(t) \in \mathbb{R}$ the input and $y_{i,k}(t) \in \mathbb{R}$ the output. Accordingly, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^n$, $C \in \mathbb{R}^{1\times n}$.

We use the notation of the time-shift operator q^{-1} as proposed by [6]. It is defined by $\mathrm{q}^{-1}x(t) = x(t-1)$, i.e., $\mathrm{q}^{-1}x(t)$ denotes the value of the signal $x(t)$ from the previous time step, $\mathrm{q}^{-2}x(t)$ the value from two time steps before, and so on. Using this, the state-space system (8) can be represented as

$$y_{i,k}(t) = P(\mathrm{q})u_{i,k}(t) + d_i(t), \tag{9}$$

where $d_i(t) = CA^t x_i(0)$ is the free response to the initial condition, which is assumed to be repeatable for every iteration. The input–output mapping $P(\mathrm{q})$ is given by an infinite power series,

$$P(\mathrm{q}) = p_1\mathrm{q}^{-1} + p_2\mathrm{q}^{-2} + p_3\mathrm{q}^{-3} + \ldots, \tag{10}$$

with Markov parameters $p_m = CA^{m-1}B$. The relative degree r of the system (8) defines the number of coefficients equal zero; that is, $p_m = 0$ for $m < r$, $p_m \neq 0$ for $m = r$. More intuitively, the relative degree is the delay of a system, i.e. the number of time steps after which an input stimulus affects and is 'visible' in the system output.

3.1.2 Error signals

The only information exchanged between agents is the relative position, which can be easily obtained in practical implementations. For example, an agent's own camera system could detect the neighbors and measure the distances; no communication between agents would be necessary.

We define an error signal $e_{i,k}(t)$ for each agent i as the mean of relative distances to all its neighbors and to the virtual leader if accessible,

$$\begin{aligned} e_{i,k}(t) = \frac{1}{d_i^{\text{in}} + b_i}\Bigg(&\sum_{j=1}^{N} a_{ij}\big(y_{j,k}(t) - y_{i,k}(t)\big) \\ &+ b_i\big(y_{des}(t) - y_{i,k}(t)\big)\Bigg). \end{aligned} \tag{11}$$

Dividing by the factor $(d_i^{\text{in}} + b_i)$ allows easier handling of different graph topologies. For example, adding additional communication edges or more agents can be done without adapting the learning gains, as we will see later.

Remark 2 The error function (11) is for the consensus case, in which all agents aim to follow the same trajectory. If the goal is a desired constant or time-varying formation, additional iteration-invariant terms, the desired distances $\Delta_{ij}(t)$ and $\Delta_i(t)$, have to be added. Thus,

$$\begin{aligned} e_{i,k}(t) = \frac{1}{d_i^{\text{in}} + b_i}\Bigg(&\sum_{j=1}^{N} a_{ij}\big(y_{j,k}(t) - y_{i,k}(t) + \Delta_{ij}(t)\big) \\ &+ b_i\big(y_{des}(t) - y_{i,k}(t) + \Delta_i(t)\big)\Bigg). \end{aligned} \tag{12}$$

3.2 Theoretical analysis of distributed iterative learning control

As ILC approach, we use the distributed input update rule

$$u_{i,k+1}(t) = u_{i,k}(t) + L(\mathrm{q})e_{i,k}(t+r), \tag{13}$$

with $i \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T-r\}$, and causal learning function [6]

$$L(\mathrm{q}) = l_0 + l_1\mathrm{q}^{-1} + l_2\mathrm{q}^{-2} + \cdots, \tag{14}$$

where r is the relative degree and the l_* values are the hyperparameters of the learning algorithm. The block diagram is shown in Fig. 3. In the following, the relative degree is assumed to be one without loss of generality. A higher relative degree can be compensated for by a corresponding time shift of the error vector (this becomes more evident with the definitons in (16). A time shift of the vector means shifting each of its elements one column up). The acausality in

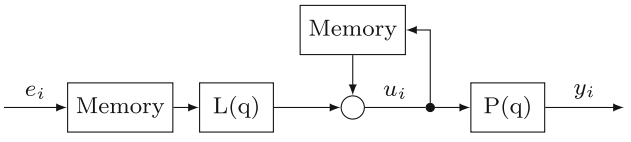

Fig. 3 Basic ILC structure for vehicle v_i. $P(\mathrm{q})$ denotes the plant, $L(\mathrm{q})$ the learning function. The input is computed based on the previous input and previous distributed tracking error e_i, which are saved in memory units

the update rule, i.e. the fact that $u_{i,k+1}(t)$ depends on future values $e_{i,k}(t+r)$, is not a problem as the input update is computed after the previous execution was completed. Algorithm (13) can be interpreted as a simple, first-order ILC algorithm commonly used in the literature, see [6]. The order of an ILC algorithm is defined as the number of previous iterations taken into account.

Multi-agent ILC was first presented in [10] and similar approaches can be found in [11,12]. They use D-type ILC algorithms, a special case of the learning function $L(\mathrm{q})$ proposed in this work.

For analyzing stability of the learning scheme, that is, convergence of the learned input as the number of iterations goes to infinity, we use the lifted-system representation according to [6], where all samples of a signal are stacked in a large vector. The system dynamics (9) and the input update (13) are now represented by

$$\underbrace{\begin{bmatrix} y_{i,k}(1) \\ y_{i,k}(2) \\ \vdots \\ y_{i,k}(T) \end{bmatrix}}_{\mathrm{y}_{i,k}} = \mathrm{P} \underbrace{\begin{bmatrix} u_{i,k}(0) \\ u_{i,k}(1) \\ \vdots \\ u_{i,k}(T-1) \end{bmatrix}}_{\mathrm{u}_{i,k}} + \underbrace{\begin{bmatrix} d_i(1) \\ d_i(2) \\ \vdots \\ d_i(T) \end{bmatrix}}_{\mathrm{d}_i}, \tag{15}$$

$$\underbrace{\begin{bmatrix} u_{i,k+1}(0) \\ u_{i,k+1}(1) \\ \vdots \\ u_{i,k+1}(T-1) \end{bmatrix}}_{\mathrm{u}_{i,k+1}} = \underbrace{\begin{bmatrix} u_{i,k}(0) \\ u_{i,k}(1) \\ \vdots \\ u_{i,k}(T-1) \end{bmatrix}}_{\mathrm{u}_{i,k}} + \mathrm{L} \underbrace{\begin{bmatrix} e_{i,k}(1) \\ e_{i,k}(2) \\ \vdots \\ e_{i,k}(T) \end{bmatrix}}_{\mathrm{e}_{i,k}}, \tag{16}$$

with lower-triangular, Toeplitz matrices

$$\mathrm{P} = \begin{bmatrix} p_1 & 0 & \dots & 0 \\ p_2 & p_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_T & p_{T-1} & \dots & p_1 \end{bmatrix},$$

$$\mathrm{L} = \begin{bmatrix} l_0 & 0 & \dots & 0 \\ l_1 & l_0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{T-1} & l_{T-2} & \dots & l_0 \end{bmatrix}.$$

We now combine all single-agent dynamics into one equation

$$\underbrace{\begin{bmatrix} \mathrm{y}_{1,k} \\ \mathrm{y}_{2,k} \\ \vdots \\ \mathrm{y}_{N,k} \end{bmatrix}}_{\mathrm{Y}_k} = \underbrace{\begin{bmatrix} \mathrm{P} & 0 & \dots & 0 \\ 0 & \mathrm{P} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \mathrm{P} \end{bmatrix}}_{\mathrm{I}_N \otimes \mathrm{P}} \underbrace{\begin{bmatrix} \mathrm{u}_{1,k} \\ \mathrm{u}_{2,k} \\ \vdots \\ \mathrm{u}_{N,k} \end{bmatrix}}_{\mathrm{U}_k} + \underbrace{\begin{bmatrix} \mathrm{d}_1 \\ \mathrm{d}_2 \\ \vdots \\ \mathrm{d}_N \end{bmatrix}}_{\mathrm{D}}, \tag{17}$$

where $\otimes$ denotes the Kronecker product and I_N the $(N \times N)$ identity matrix. Analogously, using the error definition (11), the multi-agent version of (16) is

$$\begin{aligned} \underbrace{\begin{bmatrix} \mathrm{u}_{1,k+1} \\ \mathrm{u}_{2,k+1} \\ \vdots \\ \mathrm{u}_{N,k+1} \end{bmatrix}}_{\mathrm{U}_{k+1}} &= \underbrace{\begin{bmatrix} \mathrm{u}_{1,k} \\ \mathrm{u}_{2,k} \\ \vdots \\ \mathrm{u}_{N,k} \end{bmatrix}}_{\mathrm{U}_k} + \underbrace{\begin{bmatrix} \mathrm{L} & 0 & \dots & 0 \\ 0 & \mathrm{L} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \mathrm{L} \end{bmatrix}}_{\mathrm{I}_N \otimes \mathrm{L}} \\ &\cdot \left(\underbrace{\begin{bmatrix} \frac{1}{d_1^{\mathrm{in}}+b_1} & 0 & \dots & 0 \\ 0 & \frac{1}{d_2^{\mathrm{in}}+b_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \frac{1}{d_N^{\mathrm{in}}+b_N} \end{bmatrix}}_{W} \otimes \mathrm{I}_T \right) \\ &\cdot \left\{ \left(\underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & \dots & \dots & a_{NN} \end{bmatrix}}_{\mathcal{A}} \otimes \mathrm{I}_T \right) \underbrace{\begin{bmatrix} \mathrm{y}_{1,k} \\ \mathrm{y}_{2,k} \\ \vdots \\ \mathrm{y}_{N,k} \end{bmatrix}}_{\mathrm{Y}_k} \right. \\ &- \left(\underbrace{\begin{bmatrix} d_1^{\mathrm{in}} & 0 & \dots & 0 \\ 0 & d_2^{\mathrm{in}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & d_N^{\mathrm{in}} \end{bmatrix}}_{\mathcal{D}} \otimes \mathrm{I}_T \right) \underbrace{\begin{bmatrix} \mathrm{y}_{1,k} \\ \mathrm{y}_{2,k} \\ \vdots \\ \mathrm{y}_{N,k} \end{bmatrix}}_{\mathrm{Y}_k} \\ &+ \left(\underbrace{\begin{bmatrix} b_1 & 0 & \dots & 0 \\ 0 & b_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & b_N \end{bmatrix}}_{\mathcal{B}} \otimes \mathrm{I}_T \right) \\ &\left. \cdot \left(\underbrace{\begin{bmatrix} \mathrm{y}_{des} \\ \mathrm{y}_{des} \\ \vdots \\ \mathrm{y}_{des} \end{bmatrix}}_{\mathrm{Y}_{des}} - \underbrace{\begin{bmatrix} \mathrm{y}_{1,k} \\ \mathrm{y}_{2,k} \\ \vdots \\ \mathrm{y}_{N,k} \end{bmatrix}}_{\mathrm{Y}_k} \right) \right\}, \end{aligned} \tag{18}$$

where $\mathrm{y}_{des} = \big(y_{des}(1), y_{des}(2), \ldots, y_{des}(T)\big)^T$. With the property of the Kronecker product that $(A \otimes B)(C \otimes D) = AC \otimes BD$ and the graph-theoretic definitions from Sect. 2, $\mathcal{L}_\mathcal{G} = \mathcal{D} - \mathcal{A}$ and $\widehat{\mathcal{L}} = W(\mathcal{L}_\mathcal{G} + \mathcal{B})$ from (6), this can be written as

$$\mathrm{U}_{k+1} = \mathrm{U}_k - (\mathrm{I}_N \otimes \mathrm{L}) \cdot \big((\widehat{\mathcal{L}} \otimes \mathrm{I}_T)\mathrm{Y}_k - (W\mathrm{b} \otimes \mathrm{I}_T)\mathrm{y}_{des}\big). \tag{19}$$

For simplicity, we collect all iteration-invariant terms, that is, terms only depending on the initial conditions or the reference trajectory, in $\delta \in \mathbb{R}^{NT}$. Then, by plugging (17) into (19), it follows

$$\begin{aligned}\mathrm{U}_{k+1} &= \mathrm{U}_k - (\mathrm{I}_N \otimes \mathrm{L})(\widehat{\mathcal{L}} \otimes \mathrm{I}_T)(\mathrm{I}_N \otimes \mathrm{P})\mathrm{U}_k + \delta \\ &= \mathrm{U}_k - (\mathrm{I}_N \widehat{\mathcal{L}} \mathrm{I}_N \otimes \mathrm{L}\,\mathrm{I}_T\,\mathrm{P})\mathrm{U}_k + \delta \\ &= (\mathrm{I}_{NT} - \widehat{\mathcal{L}} \otimes \mathrm{LP})\mathrm{U}_k + \delta. \end{aligned} \tag{20}$$

Based on stability theory for discrete systems, conditions for the stability of single-agent ILC were developed in [24] and, slightly modified, presented in [6]. The following definitions are taken from the latter and adapted to MAS.

Definition 1 An ILC system is called *asymptotically stable* if there exists $\overline{\mathrm{U}} \in \mathbb{R}^{NT}$ such that for all $k = \{0, 1, \ldots\}$:

$$|\mathrm{U}_k| \leq \overline{\mathrm{U}} \quad \text{and} \quad \lim_{k\to\infty} \mathrm{U}_k \quad \text{exists},$$

where the operators, $\leq$ and lim, are defined elementwise.

Using this definition, Eq. (20), and the notation of the spectral radius ρ as the maximum absolute eigenvalue, we can state:

Theorem 1 *The multi-agent ILC system* (17)–(20) *is asymptotically stable if and only if*

$$\rho\big(\mathrm{I}_{NT} - \widehat{\mathcal{L}} \otimes \mathrm{LP}\big) < 1, \tag{21}$$

or equivalently,

$$\max_i |1 - \lambda_i l_0 p_1| < 1, \tag{22}$$

where λ_i, $i \in \{1, \ldots, N\}$, are the eigenvalues of $\widehat{\mathcal{L}}$.

Note that we assumed a relative degree $r = 1$ and, thus, $p_1 \neq 0$. The eigenvalues of $\widehat{\mathcal{L}}$ depend on the graph topology and can be easily computed for a given structure. As stability depends only on l_0, the learning function $L(\mathrm{q})$ has a multitude of tuning parameters that can be chosen to achieve good convergence behavior and overall learning performance.

Proof The first statement (21) follows directly from [24] applied to (20). The latter equation (22) is obtained by applying a similarity transformation to (14) similarly to how it is done for undirected graphs in [11].

As all entries in $\widehat{\mathcal{L}}$ are real, this matrix can be transformed to Jordan normal form, that is, it exists a regular matrix $S \in \mathbb{R}^{N\times N}$ and a Jordan matrix $J \in \mathbb{R}^{N\times N}$ with eigenvalues on the diagonal and possible ones on the subdiagonal such that

$$S^{-1}\widehat{\mathcal{L}}S = J. \tag{23}$$

Usually J is defined with ones on the superdiagonal but, for simplicity, we use this less common definition to get lower-triangular matrices. As eigenvalues and, thus, the spectral radius remain the same under similarity transformations, it follows that

$$\begin{aligned}&\rho(\mathrm{I}_{NT} - \widehat{\mathcal{L}} \otimes \mathrm{LP}) \\ &\quad = \rho\big((S \otimes \mathrm{I}_T)^{-1}(\mathrm{I}_{NT} - \widehat{\mathcal{L}} \otimes \mathrm{LP})(S \otimes \mathrm{I}_T)\big) \\ &\quad = \rho(\mathrm{I}_{NT} - J \otimes \mathrm{LP}), \end{aligned} \tag{24}$$

where L and P are lower triangular matrices, as is J. As a result, the eigenvalues are the diagonal entries. Multiplication of triangular matrices does not affect this property, thus (21) is equivalent to the scalar condition (22). □

Theorem 2 *For asymptotic stability of the ILC algorithm, it is necessary that the graph $\mathcal{G}^*$ contains a spanning tree with the virtual leader as root.*

Proof It is easy to see that (22) only holds if $\lambda_i \neq 0$. With Proposition 4, the statement is proven. □

Theorem 3 *Under the condition stated in Theorem 2, the multi-agent ILC algorithm* (13) *is asymptotically stable for all possible communication topologies and any number of agents if the learning function* (14) *is chosen such that*

$$\mathrm{sgn}(l_0) = \mathrm{sgn}(p_1) \quad \textit{and} \quad |l_0| < \frac{1}{|p_1|}.$$

Proof The proof follows directly from (22) and the bounds on the eigenvalues stated in Propositions 3 and 4. □

Theorem 4 *If the linear multi-agent ILC system is asymptotically stable, all agents' output trajectories converge to the desired trajectory with increasing number of iterations. This means that all agents learn to track the reference perfectly.*

Proof From Definition 1, we know that if the ILC is asymptotically stable, it holds that, for $k \to \infty$, $\mathrm{U}_k = \mathrm{U}_{k+1}$. So it follows from (19) that

$$0 = (\mathrm{I}_N \otimes \mathrm{L}) \cdot \big((\widehat{\mathcal{L}} \otimes \mathrm{I}_T)\mathrm{Y}_\infty - (W\mathrm{b} \otimes \mathrm{I}_T)\mathrm{y}_{des}\big), \tag{25}$$

where Y_∞ denotes the converged output vector. Invertibility is guaranteed for L. From Corollary 2 and its proof, we know

that $\widehat{\mathcal{L}}$ must be full rank, too. Using the property of the Kronecker product that $(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1})$ if A, B are invertible, we get

$$\begin{aligned}(\widehat{\mathcal{L}} \otimes \mathrm{I}_T)\mathrm{Y}_\infty &= (W\mathrm{b} \otimes \mathrm{I}_T)\mathrm{y}_{des} \\ \mathrm{Y}_\infty &= (\widehat{\mathcal{L}}^{-1} W\mathrm{b} \otimes \mathrm{I}_T)\mathrm{y}_{des}. \end{aligned} \tag{26}$$

With $\mathbf{1}_N$ denoting a column vector with all entries being one, it can be easily checked that $(\mathcal{L}_\mathcal{G} + \mathcal{B})\mathbf{1}_N = \mathrm{b}$ and hence $\widehat{\mathcal{L}}^{-1} W\mathrm{b} = \mathbf{1}_N$ [22]. Finally, we end up with

$$\mathrm{Y}_\infty = (\mathbf{1}_N \otimes \mathrm{I}_T)\mathrm{y}_{des}. \tag{27}$$

We see that all agents converge to the desired trajectory. □

Remark 3 The extension of these results to quadratic multiple-input, multiple-output (MIMO) agent dynamics is straightforward. The coefficients l_0 and q_1, which are scalar in the SISO case, are matrices $l_0, q_1 \in \mathbb{R}^{\mathsf{m}\times\mathsf{m}}$. The matrix product LP then results in a block triangular matrix and its spectrum is given by the m eigenvalues e_j of the matrix product $l_0 q_1$. In the spectrum of LP, each of the eigenvalues e_j has algebraic multiplicity T. Thus, condition (22) for MIMO agent dynamics is

$$\max_{i,\mathsf{j}} |1 - \lambda_i \mathsf{e}_\mathsf{j}| < 1. \tag{28}$$

Remark 4 For linear time-varying (LTV) agent dynamics, (21) still holds, but L and P are no longer Toeplitz.

3.3 Learning convergence under dynamic agent coupling

We now assume that there is an additional feedback term in the time-domain, that can be described as a function of the relative distances between neighbors,

$$u_{i,k}^{\mathrm{FB}}(t) = C(\mathrm{q})e_{i,k}(t) \tag{29}$$

with $e_{i,k}(t)$ as in (11) and a causal feedback function

$$C(\mathrm{q}) = c_0 + c_1\mathrm{q}^{-1} + c_2\mathrm{q}^{-2} + \cdots \tag{30}$$

This could be a feedback controller or any other dynamic coupling, which does not have to be known.

Theorem 5 *Given an arbitrary feedback component in the form of* (29)*, the stability of the distributed ILC algorithm* (13) *is not affected by the feedback if applied in parallel structure, as explained in* [6] *and Fig.* 4.

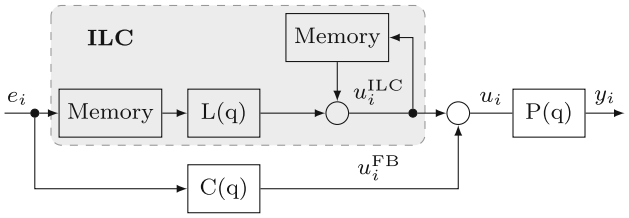

Fig. 4 Time-domain feedback control combined with iteration-domain ILC for agent v_i. The feedback term $C(\mathrm{q})$ computes updates at every time step, while the ILC part computes updates after each iteration using the past iteration's error and input signal

Proof The new input signal is $u_{i,k} = u_{i,k}^{\mathrm{ILC}} + u_{i,k}^{\mathrm{FB}}$, where $u_{i,k}^{\mathrm{ILC}}$ represents the ILC input described in Sect. 3.2. Using again the lifted-system representation and inserting into (15) yields

$$\mathrm{y}_{i,k} = \mathrm{P}\left(\mathrm{u}_{i,k}^{\mathrm{ILC}} + \mathrm{C}e_{i,k} + C_0 e_i(0)\right) + \mathrm{d}_i, \tag{31}$$

with

$$\mathrm{C} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ c_0 & 0 & \dots & 0 & 0 \\ c_1 & c_0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{T-2} & c_{T-3} & \dots & c_0 & 0 \end{bmatrix}, \mathrm{C}_0 = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{T-1} \end{bmatrix}.$$

The different sample times in the definitions of $\mathrm{e}_{i,k}$ (11) and $\mathrm{u}_{i,k}$ (13) cause the additional initial condition term $C_0 e_i(0)$ in (31) and the subdiagonal shift in C. Analogously to before, the single-agent signals can be combined into the multi-agent form

$$\begin{aligned}\mathrm{Y}_k &= (\mathrm{I}_N \otimes \mathrm{P})\left(\mathrm{U}_k^{\mathrm{ILC}} - (\mathrm{I}_N \otimes \mathrm{C})(\widehat{\mathcal{L}} \otimes \mathrm{I}_T)\mathrm{Y}_k\right) + \delta \\ &= (\mathrm{I}_N \otimes \mathrm{P})\mathrm{U}_k^{\mathrm{ILC}} - (\widehat{\mathcal{L}} \otimes \mathrm{PC})\mathrm{Y}_k + \delta \\ &= (\mathrm{I}_{NT} + \widehat{\mathcal{L}} \otimes \mathrm{PC})^{-1}(\mathrm{I}_N \otimes \mathrm{P})\mathrm{U}_k^{\mathrm{ILC}} + \delta, \end{aligned} \tag{32}$$

where all iteration-invariant terms are gathered in δ. Invertibility is guaranteed as we will see later. Inserting (32) into (19) leads to

$$\begin{aligned}\mathrm{U}_{k+1}^{\mathrm{ILC}} &= \mathrm{U}_k^{\mathrm{ILC}} - (\mathrm{I}_N \otimes \mathrm{L})(\widehat{\mathcal{L}} \otimes \mathrm{I}_T)(\mathrm{I}_{NT} + \widehat{\mathcal{L}} \otimes \mathrm{PC})^{-1} \\ &\quad \cdot (\mathrm{I}_N \otimes \mathrm{P})\mathrm{U}_k^{\mathrm{ILC}} + \delta \\ &= \underbrace{\left(\mathrm{I}_{NT} - (\widehat{\mathcal{L}} \otimes \mathrm{L})(\mathrm{I}_{NT} + \widehat{\mathcal{L}} \otimes \mathrm{PC})^{-1}(\mathrm{I}_N \otimes \mathrm{P})\right)}_{M} \\ &\quad \cdot \mathrm{U}_k^{\mathrm{ILC}} + \delta \end{aligned} \tag{33}$$

As stability is determined by the spectral radius $\rho(M)$, we must investigate the eigenvalues of this matrix. The similarity transformation (23) can be applied to get

$$\begin{aligned}\rho(M) &= \rho\Big((S\otimes \mathrm{I}_T)^{-1}\big(\mathrm{I}_{NT} - (\widehat{\mathcal{L}}\otimes \mathrm{L})(\mathrm{I}_{NT} + \widehat{\mathcal{L}}\otimes \mathrm{PC})^{-1}\\ &\quad\cdot(\mathrm{I}_N\otimes \mathrm{P})\big)(S\otimes \mathrm{I}_T)\Big)\\ &= \rho\Big(\mathrm{I}_{NT} - (S\otimes \mathrm{I}_T)^{-1}(\widehat{\mathcal{L}}\otimes \mathrm{L})(S\otimes \mathrm{I}_T)\\ &\quad\cdot(S\otimes \mathrm{I}_T)^{-1}(\mathrm{I}_{NT} + \widehat{\mathcal{L}}\otimes \mathrm{PC})^{-1}(S\otimes \mathrm{I}_T)\\ &\quad\cdot(S\otimes \mathrm{I}_T)^{-1}(\mathrm{I}_N\otimes \mathrm{P})(S\otimes \mathrm{I}_T)\Big)\\ &= \rho\Big(\mathrm{I}_{NT} - (J\otimes \mathrm{L})\big((S\otimes \mathrm{I}_T)^{-1}(\mathrm{I}_{NT} + \widehat{\mathcal{L}}\otimes \mathrm{PC})\\ &\quad\cdot(S\otimes \mathrm{I}_T)\big)^{-1}(\mathrm{I}_N\otimes \mathrm{P})\Big)\\ &= \rho\big(\mathrm{I}_{NT} - (J\otimes \mathrm{L})(\mathrm{I}_{NT} + J\otimes \mathrm{PC})^{-1}(\mathrm{I}_N\otimes \mathrm{P})\big).\end{aligned} \tag{34}$$

In (34), we inserted additional identity matrices $\mathrm{I}_{NT} = (S\otimes \mathrm{I}_T)(S\otimes \mathrm{I}_T)^{-1}$ and used the facts that $A^{-1}B^{-1} = (BA)^{-1}$ and $(A\otimes B)^{-1} = A^{-1}\otimes B^{-1}$ for two invertible matrices A and B. As all matrices are now in lower triangular form (the inverse of a triangular matrix is also triangular), the eigenvalues are the diagonal entries. As all diagonal entries of C are 0, it can be seen that the diagonal entries of $(\mathrm{I}_{NT} + J\otimes \mathrm{PC})$ are all 1. This allows us to invert the matrix and the diagonal entries of $(\mathrm{I}_{NT} + J\otimes \mathrm{PC})^{-1}$ are also all equal to one. Finally, we end up with the same condition as in Theorem 1. □

3.4 Consensus feedback controller

The above result allows us to include an additional consensus feedback controller in the form of (29). While ILC is just a feedforward control technique, that can not react to non-repeating disturbances during one iteration, this feedback controller compensates these shortcomings. This can avoid collisions between the agents, for example, and decrease the errors that are caused by non-repetitive effects. Furthermore, it enables the followers to get information even in the first iteration and, thus, reduces the initial tracking errors. This leads to better performance during the first trials.

We use the discrete-time index $t \in \{0, 1, \ldots, T\}$ to denote the sampled, continuous-time signal where $x(t)$ represents the signal value at time $t\cdot\Delta t$ with sampling interval Δt. Let the continuous-time agent dynamics be given by a double integrator

$$\ddot{y}_i(t) = u_i(t), \tag{35}$$

with output $y_i(t)\in\mathbb{R}$, input $u_i(t)\in\mathbb{R}$ and initial conditions $y_i(0) = y_{i,0}$, $\dot{y}_i(0) = \dot{y}_{i,0}$. Many systems modeled by Newtons equations can be feedback linearized resulting in independent double integrators in each direction [22].

Based on [20,22], we propose the following proportional-derivative (PD) feedback controller:

$$u_i^{\mathrm{FB}}(t) = \frac{2\eta}{\tau_c}\dot{e}_{i,k}(t) + \frac{1}{\tau_c^2}e_{i,k}(t), \tag{36}$$

with error function $e_{i,k}(t)$ as defined in (11) and controller gains, $\eta, \tau_c\in\mathbb{R}_+$.

Theorem 6 *The feedback controller given in (36) guarantees exponential stability of a multi-agent system with double-integrator agents if the communication graph contains a spanning tree. Additionally, the tracking error then converges to zero for all agents given a desired trajectory y_{des} with velocity $\dot{y}_{des} = $ constant.*

Proof As all signals are from the same iteration, we skip the iteration index k. Applying the controls (36) with (11) to the double-integrator dynamics (35), we get

$$\begin{aligned}\ddot{y}_i(t) = \frac{1}{d_i^{\mathrm{in}} + b_i}\Bigg(\frac{2\eta}{\tau_c}\Bigg(\sum_{j=1}^{N} a_{ij}\dot{y}_j(t) + b_i\dot{y}_{des}(t)\\ - (d_i^{\mathrm{in}} + b_i)\dot{y}_i(t)\Bigg) + \frac{1}{\tau_c^2}\Bigg(\sum_{j=1}^{N} a_{ij}y_j(t)\\ + b_i y_{des}(t) - (d_i^{\mathrm{in}} + b_i)y_i(t)\Bigg)\Bigg)\end{aligned} \tag{37}$$

Again, extending the single-agent dynamics to the multi-agent system, $Y(t) = (y_1(t), y_2(t), \ldots, y_N(t))^T$, leads to

$$\begin{aligned}\ddot{Y}(t) = \frac{2\eta}{\tau_c}\Big(W\mathrm{b}\dot{y}_{des}(t) - \widehat{\mathcal{L}}\dot{Y}(t)\Big)\\ + \frac{1}{\tau_c^2}\Big(W\mathrm{b}y_{des}(t) - \widehat{\mathcal{L}}Y(t)\Big),\end{aligned} \tag{38}$$

with $\mathrm{b} = (b_1, b_2, \ldots, b_N)^T$ and the positive definite, diagonal weighting matrix W as defined in (5). For proving stability, the reference signal is set to zero, $y_{des}(t)\equiv 0$, and the second-order differential equation is transformed into a first order system,

$$\begin{bmatrix}\dot{Y}(t)\\ \ddot{Y}(t)\end{bmatrix} = \underbrace{\begin{bmatrix}0_{N\times N} & \mathrm{I}_N\\ -\frac{1}{\tau_c^2}\widehat{\mathcal{L}} & -\frac{2\eta}{\tau_c}\widehat{\mathcal{L}}\end{bmatrix}}_{A_{cl}}\begin{bmatrix}Y(t)\\ \dot{Y}(t)\end{bmatrix}. \tag{39}$$

For a block matrix

$$M = \begin{bmatrix}A & B\\ C & D\end{bmatrix},$$

where A and C commute, it is known that $det(M) = det(AD - BC)$ [22]. This can be applied to the characteristic equation to solve for the eigenvalues λ of the closed-loop

dynamics matrix A_{cl},

$$det(\lambda \mathrm{I}_{2N} - A_{cl}) = det\left(\lambda^2 \mathrm{I}_N + \left(\lambda + \frac{1}{2\eta\tau_c}\right)\frac{2\eta}{\tau_c}\widehat{\mathcal{L}}\right) = 0. \tag{40}$$

It is easy to see that $\lambda = -\frac{1}{2\eta\tau_c}$ does not solve this equation, so we get

$$det\left(\lambda^2\left(\lambda + \frac{1}{2\eta\tau_c}\right)^{-1}\mathrm{I}_N + \frac{2\eta}{\tau_c}\widehat{\mathcal{L}}\right) = 0. \tag{41}$$

Following the proof in [20] and denoting the eigenvalues of $\widehat{\mathcal{L}}$ by μ_i, it holds that for each $i \in \{1, \ldots, N\}$,

$$\mu_i = -\frac{\tau_c\lambda^2}{2\eta\lambda + \frac{1}{\tau_c}}, \tag{42}$$

and hence

$$\lambda_{i_{1,2}} = \frac{1}{\tau_c}\left(-\eta\mu_i \pm \sqrt{\eta^2\mu_i^2 - \mu_i}\right). \tag{43}$$

If the graph contains a spanning tree, we know from Proposition 4 that all $\mu_i > 0$ and, thus, all $\lambda_i < 0$. Consequently, the matrix A_{cl} is Hurwitz and the first statement is proven.

To show the second one holds true, we define $E(t) = \widehat{\mathcal{L}}Y(t) - W\mathrm{b}y_{des}(t)$. Given that $\ddot{y}_{des}(t) = 0$, we know $\ddot{E}(t) = \widehat{\mathcal{L}}\ddot{Y}(t)$, so (38) can be written as

$$\begin{aligned}\widehat{\mathcal{L}}^{-1}\ddot{E}(t) &= -\frac{2\eta}{\tau_c}\dot{E}(t) - \frac{1}{\tau_c^2}E(t)\\ \Leftrightarrow\ \ddot{E}(t) &= -\frac{2\eta}{\tau_c}\widehat{\mathcal{L}}\dot{E}(t) - \frac{1}{\tau_c^2}\widehat{\mathcal{L}}E(t).\end{aligned} \tag{44}$$

Now, with the same argumentation as above, we see that $E(t) \to 0$ for $t \to \infty$. It holds that $(\mathcal{L}_\mathcal{G} + \mathcal{B})\mathbf{1}_N = \mathrm{b}$ and hence $\widehat{\mathcal{L}}^{-1}W\mathrm{b} = \mathbf{1}_N$ [22]. From the definition of $E(t)$, it then follows that $Y(t) \to \mathbf{1}_N y_{des}(t)$, which is equivalent to $y_i(t) \to y_{des}(t)$. □

3.5 Extensions for increased robustness

3.5.1 Q-filter

When applied to real systems, the proposed ILC scheme must gracefully handle non-idealities such as high-frequency noise or model uncertainties. In particular, if the system dynamics were known exactly, there would be more direct, especially feedforward, control methods that could be applied to achieve good tracking performance. Noise in the error signal leads to an adaptation in the input signal. This is undesirable and negatively affects the learning. To increase the robustness of

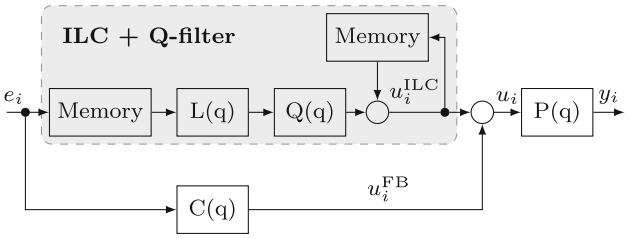

Fig. 5 Control structure with ILC and Consensus Feedback extended by a Q-filter. The output of the learning function $L(\mathrm{q})$ is filtered to reduce the influence of high-frequency noise and to guarantee smooth input signals u_i^{ILC}

ILC algorithms, it is common to include a so-called Q-filter in the input-update rule (13) as shown in Fig. 5,

$$u_{i,k+1}(t) = u_{i,k}(t) + Q(\mathrm{q})L(\mathrm{q})e_{i,k}(t+r), \tag{45}$$

with

$$Q(\mathrm{q}) = \cdots + q_{-2}\mathrm{q}^2 + q_{-1}\mathrm{q}^1 + q_0 + q_1\mathrm{q}^{-1} + q_2\mathrm{q}^{-2} + \cdots.$$

The stability conditions for the ILC algorithm with integrated Q-filter can be derived analogously to Theorem 1.

Theorem 7 *The multi-agent ILC system with Q-filter is asymptotically stable if and only if*

$$\rho\left(\mathrm{I}_{NT} - \widehat{\mathcal{L}} \otimes \mathrm{QLP}\right) < 1, \tag{46}$$

with

$$\mathrm{Q} = \begin{bmatrix} q_0 & q_{-1} & \cdots & q_{-(T-1)} \\ q_1 & q_0 & \cdots & q_{-(T-2)} \\ \vdots & \vdots & \ddots & \vdots \\ q_{T-1} & q_{T-2} & \cdots & q_0 \end{bmatrix}.$$

If the filter is causal, Q *is a lower triangular matrix and* (46) *can be reduced to the scalar inequality condition*

$$\max_i |1 - \lambda_i q_0 l_0 p_1| < 1, \tag{47}$$

where λ_i *are the eigenvalues of* $\widehat{\mathcal{L}}$.

The Q-filter can be designed to affect the learning impact of different frequency components of the error signal $e_{i,k}(t)$. For frequencies at which the gain of the filter is one, the learning is not influenced, whereas for frequencies at which the magnitude is zero, no signal is passed through the filter and those frequencies do not have an impact on the system behavior and learning. This filtering can increase robustness at the cost of poorer learning performance. For example, the stability and convergence behavior of the ILC algorithm is influenced by high frequencies in the error signal, which may

be caused by measurement noise. A theoretical analysis of these effects is given in [25]. As an adaptation to noise is undesirable, the Q-filter is implemented as a discrete lowpass filter. In practice, different types of lowpass filters can be implemented, including finite-impulse-response Gaussian filters [26] or infinite-impulse-response filters such as discrete Butterworth filters [27].

As the learning step with the input update can be computed between two iterations, all samples of the error signal are available and, thus, even non-causal filters can be easily realized. An effective method to avoid phase lags caused by the filtering is to filter the signal twice. First, the data is filtered forward in time and the result is then filtered backwards, to cancel the phase change produced in the first filtering run [25].

A good explanation of the role of the Q-filter can be found in [6]. For more insight into the trade-off between performance and robustness, see [28].

3.5.2 Kalman-filter based disturbance estimation

A more formal way to treat measurement noise and model uncertainties is by using a Kalman filter, which optimally trades off fast learning and noise rejection based on the a priori covariances provided.

The measured error $e_{i,k}$ can be divided into two parts, a repetitive share representing repeating external disturbances, modeling errors and differences in the agents' desired motions, and a non-repetitive one, which includes non-repeating disturbances and process and measurement noise. The first part is the one ILC is supposed to compensate for. The second part can be assumed to be trial-uncorrelated and Gaussian distributed with mean of zero. The idea is to use a Kalman filter approach to estimate the repetitive error component, the so-called disturbance vector d_i, as it is presented in combination with ILC in [7,29] and further studied in [30]. The block diagram is shown in Fig. 6. Note that the name d_i is chosen according to the literature, but this variable is in our case not exactly the same as the vector representing the response to the initial condition in Sects. 3.1–3.3. It additionally contains a share caused by the neighbors' movement and while they learn, this share is not fix, which in fact limits the benefit of this extension.

Error model The Kalman filter is based on a simplified model [7],

$$d_{i,k+1} = d_{i,k} + \zeta_{i,k}, \quad \zeta_{i,k} \sim \mathcal{N}(0, \mathrm{E}_k), \tag{48}$$

$$e_{i,k} = d_{i,k} + \xi_{i,k}, \quad \xi_{i,k} \sim \mathcal{N}(0, \mathrm{H}_k), \tag{49}$$

with diagonal covariance matrices $\mathrm{E}_k = \epsilon_k \mathrm{I}_T, \mathrm{H}_k = \eta_k \mathrm{I}_T$ and the initial condition for the variance of d_0, $\mathrm{P}_0 = p_0 \mathrm{I}_T$; $\epsilon_k,\ \eta_k,\ p_0 \in \mathbb{R}^+$. These stochastic values are assumed to be the same for each agent.

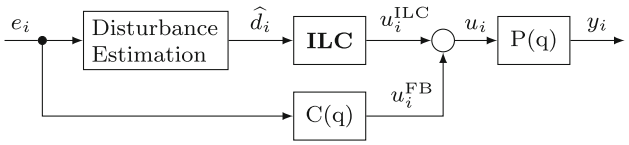

Fig. 6 Control structure with ILC and Consensus Feedback extended by an additional Disturbance Estimation. The disturbance estimation block additionally saves the estimated disturbance vector for the next iteration. Note that the ILC block holds the signals for one iteration and may include the Q-filter as depicted in Fig. 5

For a Kalman filter, a system dynamics model is usually taken into account, but this would mean that each agent has to know the whole multi-agent dynamics, what contradicts the distributed approach we are using. Moreover, since we perform learning because an accurate system model is not available, the dynamics model used for prediction is simply assuming that the disturbance is the same as in the last iteration plus noise (48). The measurement model assumes that we can measure the disturbance vector plus some white noise (49).

The estimated disturbance vector $\widehat{d}_i$, which is the basis for the input update is then computed as

$$\widehat{d}_{i,k} = \widehat{d}_{i,k-1} + \mathrm{K}_k(e_{i,k} - \widehat{d}_{i,k-1}), \tag{50}$$

with Kalman gains K_k, which are calculated recursively, according to [30],

$$\begin{aligned} \mathrm{S}_k &= \mathrm{P}_{k-1} + \mathrm{E}_{k-1} \\ \mathrm{K}_k &= \mathrm{S}_k(\mathrm{S}_k + \mathrm{H}_k)^{-1} \\ \mathrm{P}_k &= (\mathrm{I} - \mathrm{K}_k)\mathrm{S}_k. \end{aligned} \tag{51}$$

Choice of Parameters The disturbance vector $d_{i,k}$ can be viewed as the lowpass-filtered error vector, where the Kalman filter with the choice of ϵ, η optimally trades off trusting the model (that is, old disturbance = new disturbance) and trusting the measurement $e_{i,k}$. The choice of the covariance parameters ϵ_k, η_k determines the weighting between the new measured error and the old estimate of the disturbance vector, for a detailed discussion see [7]. For simplicity, we use, constant values $\epsilon_k = \epsilon$, $\eta_k = \eta$ for all k. As all matrices are diagonal, we can break down the algorithm to a scalar equation:

$$\begin{aligned} \mathrm{k}_k &= \frac{p_{k-1} + \epsilon}{p_{k-1} + \epsilon + \eta} \\ p_k &= (1 - k_k)(p_{k-1} + \epsilon), \end{aligned} \tag{52}$$

where p_k and k_k are the diagonal entries of P_k and K_k, respectively. The covariances are difficult to determine experimentally, but can be viewed as tuning parameters. In fact,

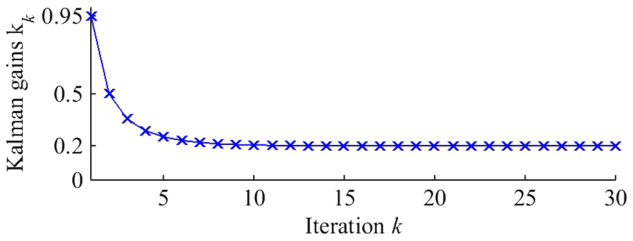

Fig. 7 Convergence behavior of Kalman gains for given $k_1 = 0.95$, $k_\infty = 0.2$ and constant covariances ϵ, η

here we choose the desired Kalman gains for the first iteration k_1 and the converged value k_∞ and solve the resulting equations for p_0, ϵ and η:

$$p_\infty = p_{\infty-1} \quad \text{(steady state)}$$

$$\Rightarrow \epsilon = \frac{k_\infty}{1 - k_\infty} p_\infty \tag{53}$$

$$k_\infty = \frac{p_\infty + \epsilon}{p_\infty + \epsilon + \eta} \tag{54}$$

$$k_1 = \frac{p_0 + \epsilon}{p_0 + \epsilon + \eta}. \tag{55}$$

Solving these equations for ϵ, η, $p_0 > 0$ and $k_1 \geq k_k \geq k_\infty$ leads to a unique solution for the Kalman gains. The latter inequalities reflect that as more information is gained less value is given to a single new measurement, while initially a fast adaptation and high k_1 are desirable.

The resulting gains for the choice of parameters $k_1 = 0.95$ and $k_\infty = 0.2$ are shown in Fig. 7.

Note that the convergence rate cannot be influenced with given k_1, k_∞ and constant covariance parameters.

4 Implementation details

To verify the practicability and effectiveness of the theoretically motivated multi-agent learning framework, we implemented the proposed algorithm on a group of quadrotors. The vehicle we used is the Parrot AR.Drone 2.0, which comes with a blackbox onboard controller. Its inputs are the commanded roll, ϕ_c, and pitch, θ_c, Euler angles, the turn rate around the vehicle's vertical axis, $\omega_{z,c}$, and a commanded velocity, $\dot{z}_c$, in global z-direction.

An overhead motion capture camera system and an appropriate state estimator provide all necessary position, velocity, attitude and rotational velocity information, and, consequently, this whole state vector s can be used for controls. For practical reasons, the ILC and feedback algorithms are implemented on a central, external computer. Nonetheless, the algorithms could all be used in a fully distributed architecture without modifications. The whole quadrotor control architecture is running on the Robot Operating System (ROS).

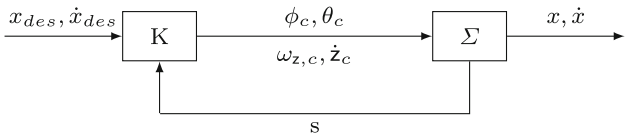

Fig. 8 Block diagram of the quadrotor control structure. The Σ block represents the quadrotor dynamics including the onboard controller with commanded roll, ϕ_c, and pitch, θ_c, Euler angles, desired turn rate around the vehicle's vertical axis, $\omega_{z,c}$ and commanded velocity, $\dot{z}_c$, as control inputs, and position x and velocity $\dot{x}$ as output. The full model state s is observable and fed back to the nonlinear position controller K, which computes the control input depending on the desired positions x_{des} and velocities $\dot{x}_{des}$

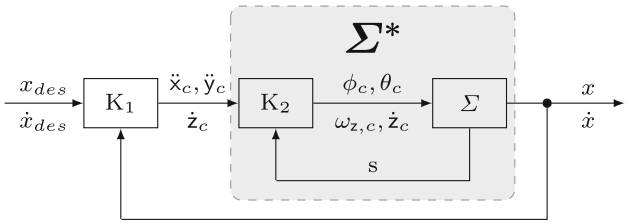

Fig. 9 Block diagram of the quadrotor control structure with controller K split up into a linear part K_1 and a nonlinear part K_2. For our theoretical analysis, the plant Σ and K_2 are summarized by system Σ^*

The quadrotors' onboard controller cannot be accessed and, therefore, can be interpreted as part of the plant Σ. Its inputs are computed by an existing, nonlinear position controller K given the desired positions $x_{des} = (x_{des}, y_{des}, z_{des})^T$ and velocities $\dot{x}_{des}$ in global coordinates as inputs. Additionally, a desired rotation matrix or desired accelerations can be set but are not used here. A sketch of the control architecture of a single vehicle is shown in Fig. 8.

The controller K can be split up into two parts, K_1 and K_2, as depicted in Fig. 9. The system K_1 compares the actual with the desired positions and velocities, and computes correcting forces/accelerations (and velocities for the z-direction) in the global coordinate system:

$$\begin{aligned}
\ddot{x}_c(t) &= \frac{2\eta}{\tau_x}\big(\dot{x}_{des}(t) - \dot{x}(t)\big) + \frac{1}{\tau_x^2}\big(x_{des}(t) - x(t)\big) \\
\ddot{y}_c(t) &= \frac{2\eta}{\tau_y}\big(\dot{y}_{des}(t) - \dot{y}(t)\big) + \frac{1}{\tau_y^2}\big(y_{des}(t) - y(t)\big) \\
\dot{z}_c(t) &= \dot{z}_{des}(t) + \frac{1}{\tau_z}\big(z_{des}(t) - z(t)\big),
\end{aligned} \tag{56}$$

where the damping ratio η and the time constants τ_x, τ_y, τ_z represent the controller gains. The resulting signals are translated into the corresponding commands ϕ_c, θ_c, $\omega_{z,c}$ and $\dot{z}_c$ in K_2. This is done using an exact input–output linearization [31], which can be applied since the camera system and a state estimator provide the full state vector s. The resulting dynamics of Σ^* can be approximated by continuous-time

double integrators decoupled for x- and y-direction. For more information about quadrotor modeling, see [32,33].

Based on this existing architecture, we can realize the consensus feedback control, as presented in Sect. 3.4, by setting the desired position and velocity to the means of the agent's neighbors' (including the virtual leader) positions and velocities,

$$\begin{aligned} \mathsf{x}_{i,des}(t) &= \frac{1}{d_i^{\text{in}} + b_i} \left(\sum_{j=1}^{N} a_{ij} \mathsf{x}_j(t) + b_i \mathsf{x}_r(t) \right) \\ \dot{\mathsf{x}}_{i,des}(t) &= \frac{1}{d_i^{\text{in}} + b_i} \left(\sum_{j=1}^{N} a_{ij} \dot{\mathsf{x}}_j(t) + b_i \dot{\mathsf{x}}_r(t) \right), \end{aligned} \tag{57}$$

where $\mathsf{x}_r(t)$ denotes the reference trajectory (virtual leader) in x-direction. The equations for y- and z-direction are defined analogously.

Furthermore, the ILC-generated feedforward input is added to the desired position. Exemplary for the x-direction, this is

$$\mathsf{x}_{i,des}(t) = \frac{1}{d_i^{\text{in}} + b_i} \Big(\sum_{j=1}^{N} a_{ij} \mathsf{x}_j(t) + b_i \mathsf{x}_r(t) \Big) + \tau_{\mathsf{x}}^2 u_{\mathsf{x},i}^{\text{ILC}}(t). \tag{58}$$

The additional factor τ_{x}^2 cancels out with the controller parameters in (56), so that the system can be transformed to a structure identical to Fig. 4.

5 Learning convergence experiments with two quadrotors

5.1 Choice of parameters

The first set of experiments focuses on motions in the horizontal plane. For these experiments in the x- and y-direction, we approximate the system dynamics by decoupled double integrators, as discussed above. The z-position is assumed to be constant. The control loop is running at a frequency $f = 66.67$ Hz. Discretization using the Taylor series expansion with time constant $\Delta t = 1/f = 0.015$s leads to the discrete dynamics

$$\begin{aligned} x_{i,k}(t+1) &= \begin{bmatrix} 1 & 0.015 \\ 0 & 1 \end{bmatrix} x_{i,k}(t) + \begin{bmatrix} \frac{1}{2}0.015^2 \\ 0.015 \end{bmatrix} u_{i,k}(t) \\ y_{i,k}(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_{i,k}(t-\tau), \end{aligned} \tag{59}$$

where τ represents the time delay of the system consisting of delays in input and output signal processing and plant-inherent delays due to the simplified modeling. As the mathematical model is linear, all these can be gathered in one delay term. We can now compute the critical parameter of the system dynamics (9)–(10), $p_1 = \frac{1}{2}0.015^2$.

For the iterative learning, a PD-type (proportional and derivative actions) input update rule is applied,

$$\begin{aligned} u_{i,k+1}^{\text{ILC}}(t) = u_{i,k}^{\text{ILC}}(t) &+ k_p e_{i,k}(t+r-1) \\ &+ k_d \frac{e_{i,k}(t+r) - e_{i,k}(t+r-2)}{2\Delta t}, \end{aligned} \tag{60}$$

with learning gains, k_p and k_d, step size Δt, and the time-shift r representing the relative degree of the plant including the delay τ. The central difference quotient is used for better noise suppression [34]. This is a special case of the learning function presented in (13) with

$$L(\mathrm{q}) = \underbrace{\frac{k_d}{2\Delta t}}_{l_0} + k_p \mathrm{q}^{-1} - \frac{k_d}{2\Delta t} \mathrm{q}^{-2}. \tag{61}$$

To determine the relative degree of the real vehicles, several effects must be taken into account including underlying dynamics from the onboard controller and motors neglected in the modeling, and system time delays mainly due to the wireless communication between the computer and the vehicle. Since these effects are difficult to measure, we identified the relative degree experimentally, e.g. by looking at the time delays in the output's open-loop response to a step-input stimulus.

The consensus feedback controller (36) is proven to be exponentially stable for double-integrator agents, see Sect. 3.4. Approximating the velocity by the difference quotient of the position, the controller can be discretized and written in the form described in (29). However, the delay term τ can destabilize the closed-loop system with feedback controller, if the graph contains cycles, see [35] or [36].

To avoid such stability issues, we set up a first experiment with a simple communication graph. The team consists of two quadrotors with agent v_1 getting information from the virtual leader, and agent v_2 only from agent v_1. Due to space limitations, it was not possible to include more agents in this experimental setup. We choose the following setup: controller parameters $\eta = 0.707$ and $\tau_c = 1.7$, ILC learning gains $k_p = 0.35$ and $k_d = 0.26$, ILC time shift $r = 49$. Assuming the time shift matches the relative degree and with the eigenvalues of the corresponding graph Laplacian $\lambda_{1,2} = 1$, we can see that (22) holds and, therefore, asymptotic stability of the ILC algorithm is guaranteed.

5.2 Experimental results

Experimental results for the ILC without the consensus feedback controller are shown in Figs. 10, 11 and 12 over 25

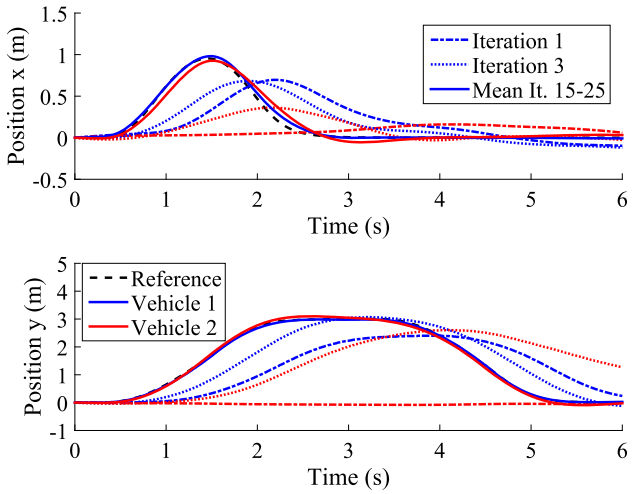

Fig. 10 Trajectories over time in x- and y-direction for the *pure ILC* algorithm. Vehicle 1 (blue) and vehicle 2 (red) learn to follow the desired reference trajectory (dashed black). Highlighted are the first (dash-dotted) and the third (dotted) iteration, and the mean over iterations 15–25 (solid) (Color figure online)

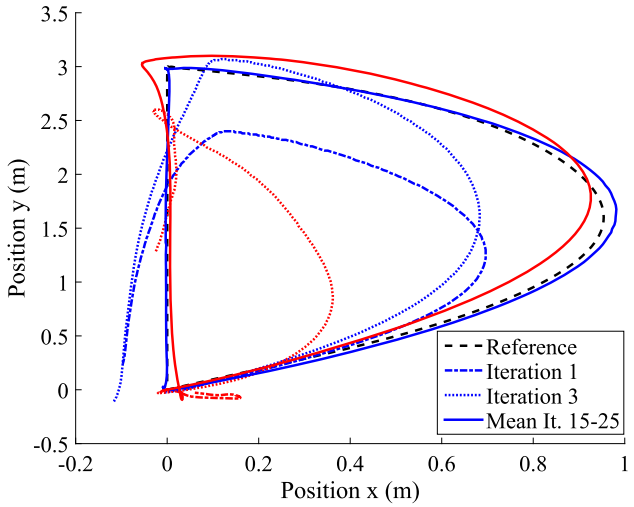

Fig. 11 Trajectories in the workspace (y over x) for the *pure ILC* algorithm. Vehicle 1 (blue) and vehicle 2 (red) significantly improve their performance with respect to the desired reference trajectory (dashed black). Highlighted are the first (dash-dotted) and the third (dotted) iteration, and the mean over iterations 15–25 (solid) (Color figure online)

iterations. Both quadrotors were flown at the same time in a given formation with a fixed distance apart. For the plots, the distance offset was subtracted. We repeated the whole learning experiment ten times and show the average error convergence and standard deviation in Figs. 12 and 17. To match the theoretic requirement of zero initial errors, we started the trajectories at the actual position of the vehicles at time $t = 0$ for each iteration and fixed the formation accordingly. For the experiments without the consensus feedback controller, each agents' individual feedback controller K as described above is enabled. This actually modifies the plant dynamics $P(\mathrm{q})$, see (59), but the crucial parameter p_1 is not affected because τ delays the feedback controller's reaction.

Figure 10 shows the position trajectories over time. It can be seen that, in the first iteration, where the input is the reference trajectory, the first vehicle (in blue) reacts delayed and shows lower amplitudes compared to the desired trajectory. The second vehicle (in red) does not follow the trajectory, as it has no reference information and the consensus feedback controller is disabled. Its motion is caused by disturbances; it reacts to its error relative to the first vehicle not until the next iteration. But after some iterations (see solid lines), both quadrotors learn to track the reference almost perfectly.

In Fig. 11, the corresponding trajectories are depicted in the xy-plane. Note that, to be precise, the goal was not to track this D-shaped trajectory but to follow the timed reference signals in Fig. 10 separately for x and y. As a consequence of improved timed trajectory tracking, the performance of both vehicles improves significantly over iterations. However, it can be seen that agent v_2 learns slower because it has no access to the desired trajectory.

The learning performance can be deduced from the convergence of the errors over iterations shown in Fig. 12. The error signal, as described in (11), of agent v_2 is computed relative to agent v_1 (solid line); that is, it describes the formation error. Accordingly, not only disturbances/noise on the second agent but also on the first one lead to relative errors. This explains the slightly higher error of agent v_2 after convergence is reached as well as the higher standard deviations. For comparison, the reference tracking error for agent v_2 is also shown, see dotted line. On average over iteration 15–20, the mean of the relative and absolute error of agent v_2 is 70% and 34% larger than that of agent v_1. The standard deviation of agent v_2's relative error is 75% larger.

In Fig. 13, we compare our multi-agent ILC approach, where the PD-type input-update rule used here is only one possibility, to the D-type algorithm in previous works [10–12]. Note that the plots show the results of a single experiment for each case. By adding a proportional gain, the learning performance improves significantly. On the one hand, the error is lower for both agents during the first iterations. The converged error (iterations 15–25), on the other hand, is similar for agent v_1, but for the second agent, it is on average almost 60% higher in the D-case. Theoretically, there would be many more tuning options for our distributed ILC input-update rule, but the focus of this work was on proving the general concepts.

We did the same learning experiments with the consensus feedback controller (58) enabled. Figure 14 shows the corresponding input trajectories. For space reasons only four seconds are plotted, while the full trajectory was six seconds. It can be seen that, initially, the ILC input is zero and the con-

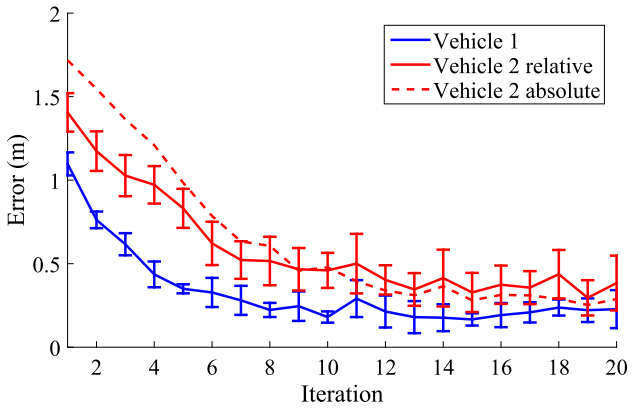

Fig. 12 Error convergence plots over iterations for the *pure ILC* algorithm. The error is computed as $\frac{1}{T}\sum_T \left(|e^{\mathsf{x}}_{i,k}(t)| + |e^{\mathsf{y}}_{i,k}(t)|\right)$ with $e_{i,k}(t)$ as in (11) for x- and y-direction, respectively. The blue and red lines are the mean errors for agent v_1 and agent v_2 over ten experiments. The bars denote the standard deviations. Note that for agent v_2, the solid red line is the error relative to its neighbor v_1 as described above, whereas the dashed line denotes the absolute error, that is, the difference to the reference trajectory (Color figure online)

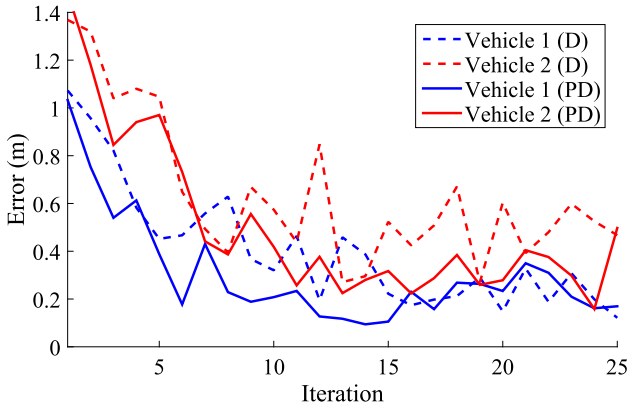

Fig. 13 Error convergence plots over iterations for *D*- versus *PD-type ILC* algorithm. The error is computed as described in Fig. 12. The blue and red lines are the errors for agent v_1 and agent v_2, respectively, of both experiments. In the first case (dashed), only a D-type ILC algorithm ($k_d = 0.26\,, k_p = 0$) is used as is common in the literature. Adding a proportional part to the learning function leads to a PD-type ILC ($k_d = 0.26\,, k_p = 0.35$) and improves the learning performance for both agents (solid lines) (Color figure online)

sensus feedback component dominates the input composed by $u_{i,k} = u^{\mathrm{ILC}}_{i,k} + u^{\mathrm{FB}}_{i,k}$. The consensus feedback also causes agent v_2 to react to the first agent already in the first iteration. But this reaction is lower in amplitude and delayed compared to the feedback signal guiding agent v_1. In contrast, after convergence is reached, the feedback input is nearly zero for both vehicles and mainly compensates for non-repetitive errors (as indicated by trajectories that vary around zero for subsequent iterations), while the ILC feedforward input compensates for the repetitive error. Comparing the converged ILC input with the initial, purely feedback-based input shows that ILC

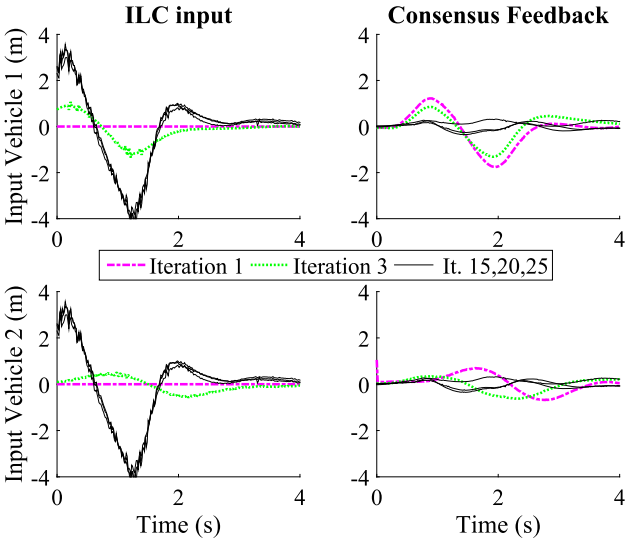

Fig. 14 Input trajectories for ILC with underlying consensus feedback controller in x-direction for vehicle 1 (top) and vehicle 2 (bottom). The ILC-generated feedforward input (left) increases over iterations to compensate for the learned repetitive disturbances, while the input from the consensus feedback controller (right) decreases and only accounts for non-repetitive disturbances at the end. Highlighted are the first (dash-dotted, magenta), the third (dotted, green) iteration and, exemplary for the converged case, iterations 15, 20 and 25 (solid, black). It can be seen that the ILC inputs are subject to high-frequency noise. For space reasons only the first 4 seconds are shown (Color figure online)

causes larger input magnitudes with peaks being time-shifted to the left. Instead of being reactive, the ILC is proactive and sends aggressive input signals that keep the vehicles on track.

Additionally, the plots of the ILC inputs in Fig. 14 show high-frequency oscillations caused by noise in the system that can affect the learning performance as described in Sect. 3.5.1. To compensate for this, a Q-filter is implemented as two discrete butterworth filters, forward and then backwards in time, to get zero phase shift. We chose a filter order of 6 and 3 Hz as cutoff frequency. This smoothens the input trajectory as can be seen in Fig. 15.

The Q-filter compensates for effects such as measurement or process noise but it cannot decrease the influence of low-frequency, non-repetitive errors on the learning. An example of these errors would be air blasts, which occur in our setup due to other vehicles moving or hovering. For this reason we extended our algorithm by a Kalman filter to estimate the repetitive share of the error, see Sect. 3.5.2. With the presented choice of parameters, $\mathrm{k}_1 = 0.95$ and $\mathrm{k}_\infty = 0.2$, the Kalman filter acts like a lowpass filter on the error signal over iterations. This means that an air blast in one iteration which moved the vehicle to the right, can be filtered out to some degree, whereas, without the Kalman extension the ILC would overreact to this disturbance and cause a shift to the left in the following iteration. The Kalman filter also reduces high-frequency oscillations in the input signal, so that an

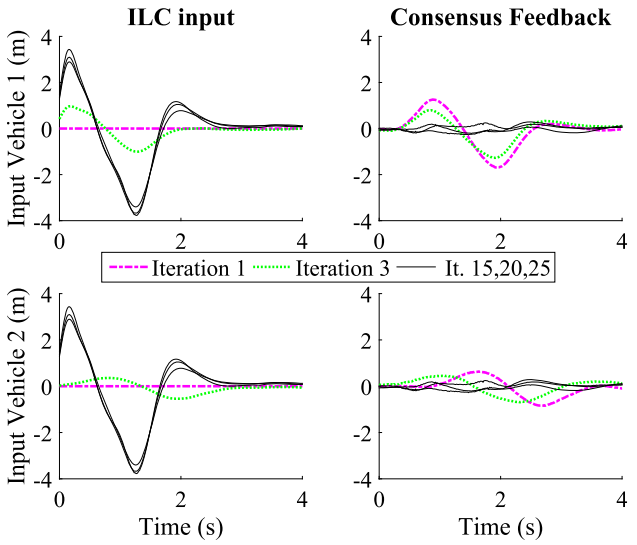

Fig. 15 Input trajectories for ILC with Q-filter and underlying consensus feedback controller in x-direction for vehicle 1 (top) and vehicle 2 (bottom). The Q-filter is implemented as a discrete Butterworth filter of order 6 with cutoff frequency $\omega_c = 3$. Compared to Fig. 14, the ILC-generated feedforward input trajectories (left) are significantly smoother (Color figure online)

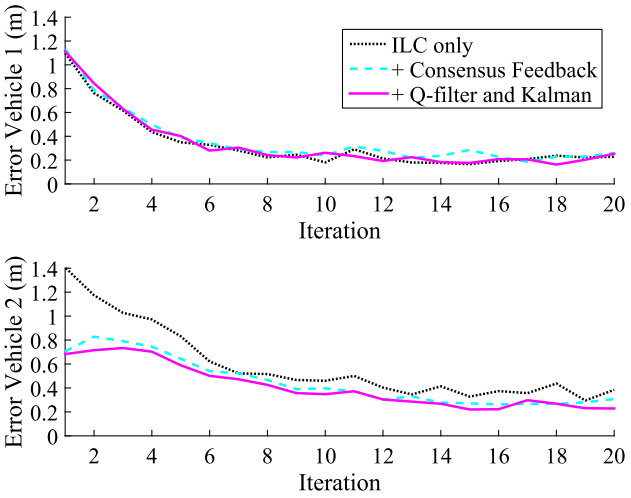

Fig. 16 Comparison of error convergence plots over iterations. The black dotted line shows the pure ILC case as presented in Fig. 12. Adding the consensus feedback controller (light blue, dashed), the initial performance and the converged relative error for vehicle 2 (bottom) significantly improve, whereas for vehicle 1 (top) the converged error slightly increases. In the third case, ILC with Consensus Feedback, Q-filter and Kalman disturbance estimation (magenta, solid), this deterioration is compensated for vehicle 1 and the learning performance for the second vehicle improves even more. All values are means over 10 experiments (Color figure online)

additional Q-filter may not be necessary. However, we used both filters at the same time to increase robustness as we did not experience a significant performance decrease caused by the Q-filter.

In Fig. 16, the error convergence plots of the position errors are shown for the three cases: pure ILC, ILC with Consensus Feedback, and ILC with Consensus Feedback and additional Q- and Kalman filters. All experiments are run over 20 iterations and repeated ten times. The plots show the mean values of the relative errors over all experiments for increased statistical reliability. Especially for the second vehicle, the learning performance improves clearly with the proposed extensions. Regarding the first iterations, enabling the consensus feedback controller guarantees that agent v_2 follows its neighbor and, thus, its error, which can be interpreted as the formation error, is significantly lower.

For a better comparison of the error after convergence is reached (iteration 15–20), the mean values and the standard deviations over 60 datasets for each case (6 iterations $\times$ 10 repetitions) are shown in Fig. 17. The error and standard deviation of agent v_1 increase when the consensus feedback controller is added to the ILC. We assume this is caused by negative effects of the time delay in the feedback loop. However, the values for the second agent decrease by 24% for the mean and 53% for the standard deviation with consensus feedback enabled. Adding the Q-filter and the Kalman estimation improves the performance even more. Compared to the case with consensus feedback, the mean of the errors decrease by 15% for agent v_1 and 11% for agent v_2. As a result, the distributed feedback has a positive impact on the performance of formation flying as it does both (i) reducing the formation error in the first iterations, which can help to avoid collisions, and (ii) accounting for non-repetitive, relative disturbances during iterations, which reduces the tracking error after learning convergence. The Q- and Kalman filtering, on the other hand, enhance the ILC algorithm itself and make it more robust against non-repetitive errors.

6 Four-drone learning under different communication graphs

To prove the effectiveness of our algorithm for a larger team of agents and the impact of varying communication structures, a different experimental setup is chosen. Instead of flying in a horizontal plane in x- and y-direction, the desired trajectory is now a simple vertical motion. This allows us to increase the team size to four quadrotors without too much (mostly non-repetitive) wind disturbance between them while hovering. As the dynamics in z-direction are faster than the horizontal motions, it was necessary to increase the control loop rate to a frequency $f = 100$ Hz. Another consequence is that the overall time delay is smaller and, as a result, we are able to use the consensus feedback controller for all graphs given below without instabilities caused by time delays.

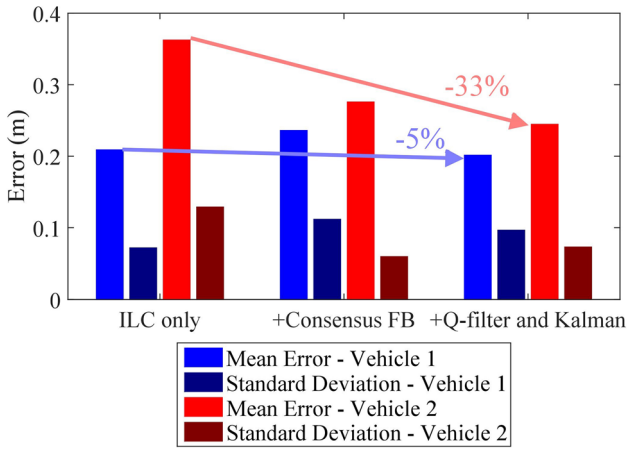

Fig. 17 Comparison of position errors for the experiments with pure ILC, with the distributed consensus feedback controller and additionally with the Q- and Kalman filters. The blue bars denote agent v_1, the red ones agent v_2 (relative error). The light bars are the mean values of the error after convergence (iterations 15–20) over ten experiments, the dark bars the corresponding standard deviations, see Fig. 12. The relative error between vehicle 1 and 2 (see red bars) significantly improves when adding the consensus controller. The best performance can be achieved by further adding the Q-filter and the Kalman approach (Color figure online)

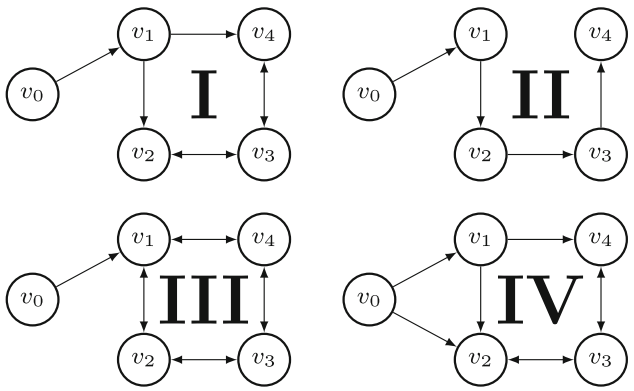

Fig. 18 Different communication graphs for the experiments in z-direction with four quadrotors. The arrows denote the information flow between the agents v_1–v_4. The reference trajectory is implemented as the virtual leader v_0

6.1 Communication graph structures

From Theorem 2, we know that the ILC is stable only if the graph contains a spanning tree, where the virtual leader node is the root. Based on this theoretical result, we choose four different communication graphs as depicted in Fig. 18.

In Graph I, agent v_1 is the team leader, who can access the reference and does not react to other vehicles, and the other three agents are getting information from both their neighbors. The Laplacian $\mathcal{L}_\mathcal{G}$ and the corresponding matrix $\widehat{\mathcal{L}}$, see (6), are

$$\mathcal{L}_\mathrm{I} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix},$$
$$\widehat{\mathcal{L}}_\mathrm{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.5 & 1 & -0.5 & 0 \\ 0 & -0.5 & 1 & -0.5 \\ -0.5 & 0 & -0.5 & 1 \end{bmatrix}. \tag{62}$$

To analyze the stability of the distributed ILC framework, the eigenvalues of $\widehat{\mathcal{L}}$ are important. For the chosen graph structures, we get the following sets of eigenvalues,

$$\lambda(\widehat{\mathcal{L}}_\mathrm{I}) = \{0.29, 1, 1, 1.71\}, \quad \lambda(\widehat{\mathcal{L}}_\mathrm{II}) = \{1, 1, 1, 1\},$$
$$\lambda(\widehat{\mathcal{L}}_\mathrm{III}) = \{0.09, 1, 1, 1.91\}, \quad \lambda(\widehat{\mathcal{L}}_\mathrm{IV}) = \{0.35, 1, 1, 1.65\}. \tag{63}$$

Note that these eigenvalues can have imaginary parts for certain other graph structures, leading to oscillatory behavior and worse performance for both, the consensus controller and the ILC. In this work, we focus on a setup where the graph structure can be chosen beforehand. Real eigenvalues can be guaranteed for graphs such as spanning trees or undirected graphs, where each pair of neighbors is exchanging information in both directions. More information can be found in [23], where the correlation of graph structures and eigenvalues is investigated.

As proven theoretically, we see that all eigenvalues are positive and can be upper-bounded by 2. With our result from Theorem 3, we can find a learning function $L(\mathrm{q})$ that guarantees stability for all possible graph structures based on the knowledge of the first Markov parameter of the single-agent plant.

6.2 Choice of parameters

In general, the dynamics in z-direction only are easier to understand than the complete quadrotor dynamics. Basically, it can be modeled as a point mass where one force is applied. This force is the sum of the four equal rotor forces minus the gravitational force, which is acting in the opposite direction. The point mass system can be described by double-integrator dynamics,

$$\ddot{\mathrm{z}} = u - g, \tag{64}$$

where g is the gravitational constant and u the mass-normalized collective thrust of the motors. As outlined in Sect. 4, the onboard controller tracks a desired velocity $\dot{\mathrm{z}}_c$. We assume there is an underlying proportional controller,

$$u = \kappa(\dot{\mathrm{z}}_c - \dot{\mathrm{z}}) + g, \tag{65}$$

where the controller gain $\kappa > 0$ is not known exactly. With the outer feedback controls (56), the closed loop dynamics for z become

$$\ddot{z} = \kappa(\dot{z}_{des} - \dot{z}) + \frac{\kappa}{\tau_z}(z_{des} - z). \tag{66}$$

Hence, we end up with second-order dynamics which can be treated in the same way as before for x and y.

The consensus feedback controller is applied as in (57). With the feedforward ILC input included analogously to before,

$$z_{i,des} = \frac{1}{d_i^{\text{in}} + b_i}\left(\sum_{j=1}^{N} a_{ij} z_j(t) + b_i z_r(t)\right) + \tau_z u_i^{\text{ILC},z}, \tag{67}$$

where $\tau_z = 1$ is chosen. Note that the Markov parameter p_1, which is crucial for the stability analysis of the ILC algorithm, cannot be computed exactly because we do not know κ exactly. Nevertheless, our PD-type ILC approach is not model-based, so it can be implemented without the exact knowledge of κ. Instead, the parameters of the learning function are adapted by conservative tuning.

For the learning gains, we chose $k_p = 1$, $k_d = 1$, and the ILC time shift $r = 13$. Additionally, a Q-filter is implemented in the same way as before as two discrete Butterworth filters with cutoff frequencies at 3 Hz and filter order 5. As the dynamics in z are faster and thus more sensitive to noise, the filtering was crucial for this experiment. On the other hand, the faster dynamics allowed for this more aggressive tuning of the learning hyperparameters. The Kalman disturbance estimation is not used here, because the different graph structures lead to large variations in the learning convergence rate. This would make it necessary to adapt the convergence rates of the Kalman filter for every single agent and communication graph.

6.3 Experimental results

We begin with graph I and show exemplary how the ILC improves the reference tracking and the synchronicity of the quadrotor team after several trials. For the other graphs in Fig. 18, we focus on the error convergence and their advantages and disadvantages. In all setups, the team forms a rectangle in the xy-plane correlating to the communication graph.

6.3.1 Graph I

This communication structure describes the case where one agent, v_1, is the team leader and the others are followers. The leader has access to the reference information and due to its special position, it is supposed to not be influenced by the other vehicles. The followers, $v_2 - v_4$, do not necessarily know who is the leader. So they react to both their neighbors.

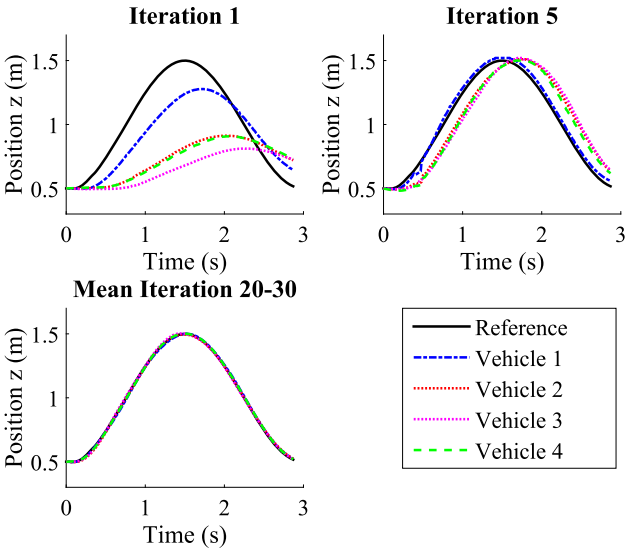

Fig. 19 Trajectories over time in z-direction for the ILC experiment with four drones communicating as depicted in Graph I in Fig. 18. In the first iteration, the consensus feedback controller causes vehicle 1 (blue, dash-dotted) to react directly to the reference trajectory (black, solid), but delayed and with lower amplitude. Vehicles 2 (red, dotted) and 4 (green, dashed) are affected by the movement of the first vehicle, but are also waiting for vehicle 3 (magenta, dotted), which follows them. After five iterations the learning improved the performance significantly. The first vehicle tracks the reference well and the others are moving synchronously. Finally, the mean over iteration 20–30 shows perfect tracking for the whole group of quadrotors (Color figure online)

Figure 19 shows the trajectories for several iterations. The reference trajectory (solid black) is a feasible, nearly sinusoidal motion starting at 0.5 m, going up to 1.5 m and back to the initial position. In the first iteration, only the consensus feedback controller is active. This causes concatenated delayed reactions of agent v_1 (blue), followed by v_2 (red) and v_4 (green) and finally v_3 (magenta) only reacting to the two latter ones. The same can be seen for the amplitudes, v_1 is affected by the desired trajectory to 100%, whereas the others try to move towards the mean of their neighbors' positions. After five iterations, the leader has learned to track the reference very closely and also the followers are tracking the reference better and almost synchronously. The third plot shows the mean of the trajectories of iteration 20–30 to average out noise and non-repeating disturbances. It can be seen, that the distributed ILC scheme enables all agents to perform the desired motion.

The corresponding error convergence plots are depicted in Fig. 20. Two different errors are shown for each vehicle, the relative error (dotted line), defined by the distance to the average of its neighbors' positions, and the absolute reference tracking error (solid). For agent v_1 both errors are identical

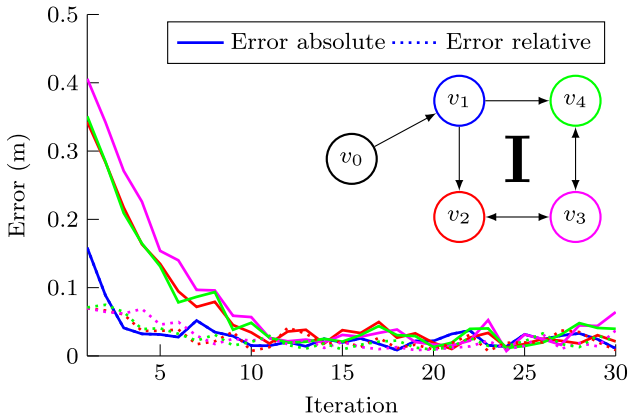

Fig. 20 Error convergence plots over iterations for Graph I. The absolute error (solid) converges fast for the leader, vehicle 1 (blue), and slower for the followers, vehicle 2 (red), 3 (magenta) and 4 (green). The relative errors (dotted) are lower due to the respective error definition (Color figure online)

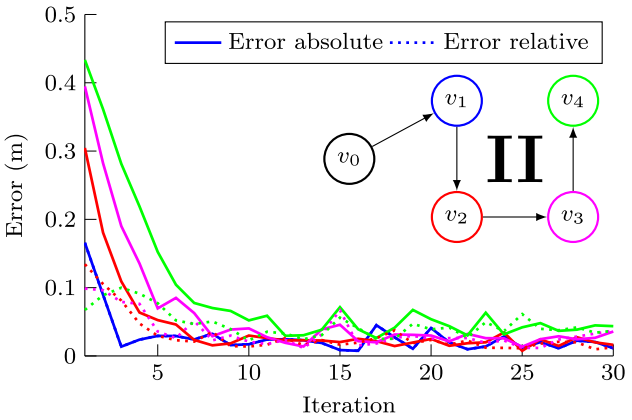

Fig. 21 Error convergence plots over iterations for Graph II. Vehicle 1 (blue) can access the desired trajectory and is followed by vehicle 2 (red). Vehicle 2 is followed by vehicle 3 (magenta), and this is followed by vehicle 4 (green). The absolute errors (solid) decrease in the same order, whereas the relative errors (dotted) are initially highest for v_1 and lowest for v_4 (Color figure online)

as the virtual leader, representing the desired trajectory, is its only neighbor. This agent's lower initial error and faster learning, as described above, can be seen in the error convergence plot, too. The plots for agents v_2 and v_4 are almost the same due to the graph's symmetry. In theory, they would behave identically; in practice, disturbances, sensor noise and slightly different physical properties of the quadrotors cause deviations. From the absolute error plots, it follows that convergence is reached after eleven iterations. For the followers, the relative errors are clearly lower than the absolute errors due to their corresponding definitions. The relative error is the difference to the average position of the neighbors and, if they are, for example, all moving in a similar way slower than the reference, this error is lower than the absolute one. However, for tasks where the focus is on synchronicity and formation flight, this error definition may be more important.

6.3.2 Graph II

In the second case, the graph is a plain spanning tree; moreover, it is a straight chain, where each vehicle only gets information from its direct predecessor, see Fig. 21. Agent v_1 behaves exactly as in the experiment before, v_2 converges faster because it is only influenced by the error relative to agent v_1. Agents v_3 and v_4 react even more delayed, but the overall learning is very fast. Convergence is reached after around seven iterations. Comparing the errors after convergence is reached, it can be seen that the average absolute error for agent v_3 and especially for v_4 are higher than for Graph I. This is a disadvantage of this chain structure. Each vehicle is influenced by the non-repetitive disturbances of all its predecessors. Furthermore, they must converge before any follower can converge. Also the relative errors, which are related to the formation error, are higher during the first iterations than they were with Graph I.

6.3.3 Graph III

Graph III is the undirected case, where information flows in both directions on each edge. Only agent v_1 is additionally influenced by the reference trajectory. As can be seen in Fig. 22, this structure ensures very low relative errors. While this guarantees a good synchronicity and low formation errors even during the first iterations, it causes slow convergence to the desired trajectory. This graph structure may be used in cases where it is more important to hold the formation (even in the first trials) than to achieve fast learning. In contrast to Graph I, agent v_1 now "waits" for the others.

6.3.4 Graph IV

Graph IV is the same as Graph I, but also agent v_2 has access to the reference now (Fig. 23). This improves the absolute error convergence for this agent as well as for agent v_3, which is exchanging information with v_2. Also the error for agent v_4, which is communicating with v_3, is improving slightly. Agent v_1 is not influenced by the behavior of the others. The peaks in the errors of vehicle 3 and 4 that occurred in the later iterations in this experiment are due to disturbances and bear no meaning.

6.3.5 Summary

Finally, we can conclude that there is a trade-off between learning performance (fast convergence) and formation syn-

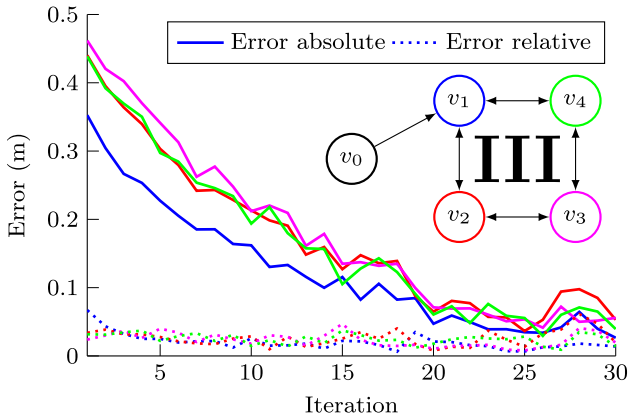

Fig. 22 Error convergence plots over iterations for Graph III. In this undirected case, the absolute errors (solid) decrease slower than for the two graphs before, with vehicle 1 (blue) converging faster than vehicle 2 (red), 3 (magenta) and 4 (green). The relative errors (dotted), on the other hand, are on a very low level. That is, the relative error is consistently kept low at the expense of a slow learning convergence (Color figure online)

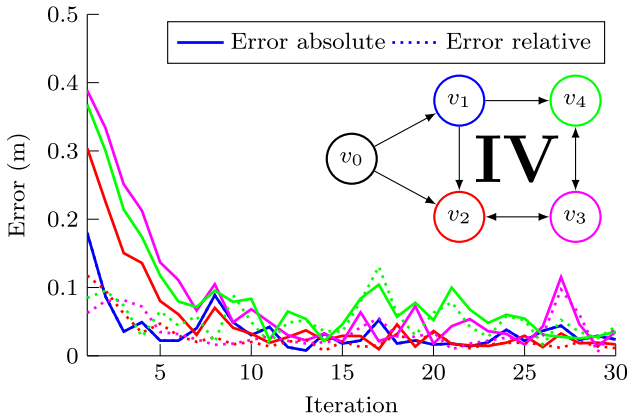

Fig. 23 Error convergence plots over iterations for Graph IV. The additional reference information access for vehicle 2 (red) decreases the absolute errors (solid) for vehicle 2 (red), 3 (magenta) and even slightly for 4 (green). Vehicle 1 (blue) is not influenced by the other vehicles' behavior. The relative errors (dotted) are similar as for Graph I, only for vehicle 2 slightly higher due to the influence of the additional edge (Color figure online)

chronicity. No bidirectional interaction as in Graph II guarantees fast convergence at the cost of higher formation errors. Adding more edges and bidirectional information exchange enables the vehicles to better hold the desired formation but slows down the adaptation to the reference. As expected, giving more agents access to the desired trajectory improves the performance. Depending on the tasks and the aims, one can choose different graph structures.

7 Conclusion

We developed a distributed ILC algorithm for multi-agent systems, which allows for arbitrary, linear and causal, learning functions. As a result, we were able to consider a PD-type input update rule extending previous work found in literature that was restricted to learning functions depending only on the error derivative (D-type). The proposed approach leads to better tracking performance and lower errors. Furthermore, many other learning function options are possible. We theoretically derived a simple scalar condition for stability of the proposed learning algorithm, independent of the number of agents or the communication structure, only assuming that the graph includes a spanning tree. However, to achieve good learning performance in practice, parameter tuning in simulations and experiment was necessary.

As ILC only compensates for repetitive disturbances, we included a consensus-based feedback controller to enable the agents to react to non-repeating disturbances and to perform better during the first iterations. That this feedback controller does not affect stability of the ILC algorithm was shown analytically. Moreover, it was shown that the same holds for any linear dynamic coupling between neighboring agents.

To increase the learning robustness against sensor or process noise and other non-repetitive disturbances, a Q-filter was included. This lowpass filter also ensures smooth ILC feedforward input trajectories and turned out to be essential for some experiments. Optionally, we presented a disturbance estimation based on a simple Kalman filter which showed to lead to faster learning and decrease the influence of non-repeating disturbances on the ILC.

In experiments with two quadrotors flying a horizontal motion, we showed that the proposed, distributed ILC algorithm achieves better reference tracking and a lower formation error for a team of quadrotors. Furthermore, the distributed consensus feedback controller decreases the influence of non-repetitive disturbances during each iteration and thus leads to a better overall formation tracking performance. The results can be further improved by including the Q-filtering and the Kalman disturbance estimation.

To show the behavior for different communication graph structures, four vehicles were flown in z-direction. To account for the different physical dynamics compared to the xy-motion, we adjusted the controller and learning gains as well as the parameters for the Q-filter, which was necessary in this setup. We concluded that there is a trade-off between the ILC learning rate and the synchronicity of the formation during each iteration, especially during the early trials. More communication edges and bidirectional information flow improve the agents' ability to wait for each other and, thus, cause a better formation synchronicity. However, this decreases the influence of the virtual leader and, therefore, slows down the adaptation to the reference trajectory.

Finally, the key contributions of this work were: a extension and generalization of the existing distributed ILC algorithms, the incorporation of a consensus-based feedback scheme for better formation holding during the learning, and

the very first experiments of distributed ILC on up to four agents.

References

1. Ren, W., Beard, R. W., & Atkins, E. M. (2005). A survey of consensus problems in multi-agent coordination. In *Proceedings of the American control conference (ACC)* (pp. 1859–1864).
2. Olfati-Saber, R., & Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, *49*(9), 1520–1533.
3. Ren, W., Beard, R. W., & Atkins, E. M. (2007). Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, *27*(2), 71–82.
4. Xie, G., & Wang, L. (2007). Consensus control for a class of networks of dynamic agents. *International Journal of Robust and Nonlinear Control*, *17*(10–11), 941–959.
5. Arimoto, S., Kawamura, S., & Miyazaki, F. (1984). Bettering operation of robots by learning. *Journal of Robotic Systems*, *1*(2), 123–140.
6. Bristow, D. A., Tharayil, M., & Alleyne, A. G. (2006). A survey of iterative learning control. *IEEE Control Systems*, *26*(3), 96–114.
7. Schoellig, A. P., Mueller, F. L., & D'Andrea, R. (2012). Optimization-based iterative learning for precise quadrocopter trajectory tracking. *Autonomous Robots*, *33*(1–2), 103–127.
8. Hehn, M., & D'Andrea, R. (2014). A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers. *Mechatronics*, *24*(8), 954–965.
9. Barton, K. L., & Alleyne, A. G. (2008). A cross-coupled iterative learning control design for precision motion control. *IEEE Transactions on Control Systems Technology*, *16*(6), 1218–1231.
10. Ahn, H.-S., & Chen, Y. (2009). Iterative learning control for multi-agent formation. In *Proceedings of ICROS-SICE international joint conference* (pp. 3111–3116).
11. Yang, S., Xu, J.-X., & Huang, D. (2012). Iterative learning control for multi-agent systems consensus tracking. In *Proceedings of the IEEE conference on decision and control (CDC)* (pp. 4672–4677).
12. Meng, D., Jia, Y., Du, J., & Yu, F. (2012). Tracking control over a finite interval for multi-agent systems with a time-varying reference trajectory. *Systems & Control Letters*, *61*(7), 807–818.
13. Liu, Y., & Jia, Y. (2012). An iterative learning approach to formation control of multi-agent systems. *Systems & Control Letters*, *61*(1), 148–154.
14. Li, J., & Li, J. (2014). Adaptive iterative learning control for coordination of second-order multi-agent systems. *International Journal of Robust and Nonlinear Control*, *24*, 3282–3299.
15. Liu, Y., & Jia, Y. (2015). Robust formation control of discrete-time multi-agent systems by iterative learning approach. *International Journal of Systems Science*, *46*(4), 625–633.
16. Meng, D., Jia, Y., & Du, J. (2015). Robust iterative learning protocols for finite-time consensus of multi-agent systems with interval uncertain topologies. *International Journal of Systems Science*, *46*(5), 857–871.
17. Yang, S., Xu, J.-X., Huang, D., & Tan, Y. (2014). Optimal iterative learning control design for multi-agent systems consensus tracking. *Systems & Control Letters*, *69*, 80–89.
18. Ahn, H.-S., Moore, K. L., & Chen, Y. (2010). Trajectory-keeping in satellite formation flying via robust periodic learning control. *International Journal of Robust and Nonlinear Control*, *20*(14), 1655–1666.
19. Hock, A., & Schoellig, A. P. (2016). Distributed iterative learning control for a team of quadrotors. In *Proceedings of the IEEE conference on decision and control (CDC)*. arXiv:1603.05933.
20. Mesbahi, M., & Egerstedt, M. (2010). *Graph theoretic methods in multiagent networks*. Princeton: Princeton University Press.
21. Bauer, F. (2012). Normalized graph Laplacians for directed graphs. *Linear Algebra and its Applications*, *436*(11), 4193–4222.
22. Ren, W., & Beard, R. W. (2008). *Distributed consensus in multi-vehicle cooperative control*. Berlin: Springer.
23. Fax, J., & Murray, R. (2004). Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, *49*(9), 1465–1476.
24. Norrlöf, M., & Gunnarsson, S. (2002). Time and frequency domain convergence properties in iterative learning control. *International Journal of Control*, *75*(14), 1114–1126.
25. Longman, R. W. (2010). Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, *73*(10), 930–954.
26. Bristow, D. A., & Alleyne, A. G. (2003). A manufacturing system for microscale robotic deposition. In *Proceedings of the American control conference (ACC)* (pp. 2620–2625).
27. Elci, H., Longman, R. W., Juang, J.-N., Ugoletti, R., Phan, M., Juang, J.-N., & Ugoletti, R. (1994). Discrete frequency based learning control for precision motion control. In: *Proceedings of the IEEE international conference on systems, man, and cybernetics* (pp. 2767–2773).
28. de Roover, D. (1996). Synthesis of a robust iterative learning controller using an H_∞ approach. In: *Proceedings of the IEEE conference on decision and control (CDC)* (pp. 3044–3049).
29. Mueller, F. L., Schoellig, A. P., & D'Andrea, R. (2012) Iterative learning of feed-forward corrections for high-performance tracking. In *Proceedings of the IEEE international conference on intelligent robots and systems* (pp. 3276–3281).
30. Degen, N., & Schoellig, A. P. (2014) Design of norm-optimal iterative learning controllers: The effect of an iteration-domain Kalman filter for disturbance estimation. In *Proceedings of the IEEE conference on decision and control (CDC)* (pp. 3590–3596).
31. Zhou, Q. L., Zhang, Y., Rabbath, C. A., & Theilliol, D. (2010). Design of feedback linearization control and reconfigurable control allocation with application to a quadrotor UAV. In: *Proceedings of the IEEE conference on control and fault tolerant systems* (pp. 371–376).
32. Michael, N., Mellinger, D., Lindsey, Q., & Kumar, V. (2010). The GRASP multiple micro-UAV testbed. *IEEE Robotics and Automation Magazine*, *17*(3), 56–65.
33. Schoellig, A. P., Hehn, M., Lupashin, S., & D'Andrea, R. (2011). Feasiblity of motion primitives for choreographed quadrocopter flight. In: *Proceedings of the American control conference (ACC)* (pp. 3843–3849).
34. Chen, Y., & Moore, K. L. (2002). An optimal design of PD-type iterative learning control with monotonic convergence. In: *Proceedings of the IEEE international symposium on intelligent control* (pp. 55–60).
35. Münz, U., Papachristodoulou, A., & Allgöwer, F. (2008). Delay-dependent rendezvous and flocking of large scale multi-agent systems with communication delays. In: *Proceedings of the IEEE conference on decision and control (CDC)* (pp. 2038–2043).
36. Hu, J., & Lin, Y. (2010). Consensus control for multi-agent systems with double-integrator dynamics and time delays. *IET Control Theory & Applications*, *4*(1), 109–118.

Andreas Hock received his B.Sc. and M.Sc. degrees in Engineering Cybernetics from the University of Stuttgart in 2013 and 2016, respectively. During his Masters, he spent a research semester at the University of Toronto in the Dynamic Systems Lab. For his studies, he received two scholarships, the Baden-Württemberg-STIPENDIUM and the Deutschlandstipendium supported by the Robert Bosch GmbH. He has previously worked in the R&D department of the Dr. Ing. h.c. F. Porsche AG, where he developed advanced feedforward trajectories for powertrain control. Currently, he is working for the Robert Bosch GmbH in the field of assisted and automated driving.

Angela P. Schoellig is an Assistant Professor at the University of Toronto Institute for Aerospace Studies and an Associate Director of the Centre for Aerial Robotics Research and Education. She holds a Canada Research Chair in Machine Learning for Robotics and Control, is a principal investigator of the NSERC Canadian Robotics Network, and is a Faculty Affiliate of the Vector Institute for Artificial Intelligence. She conducts research at the intersection of robotics, controls, and machine learning. Her goal is to enhance the performance, safety, and autonomy of robots by enabling them to learn from past experiments and from each other. She is a recipient of a Sloan Research Fellowship and an Ontario Early Researcher Award, and is one of MIT Technology Review's Innovators Under 35 (all in 2017). Her PhD at ETH Zurich (2013) was awarded the ETH Medal and the Dimitris N. Chorafas Foundation Award. She holds both an M.Sc. in Engineering Cybernetics from the University of Stuttgart (2008) and an M.Sc. in Engineering Science and Mechanics from the Georgia Institute of Technology (2007).