A Decentralized Optimal Iterative Learning Control Approach With Efficient Computation for Collaborative Tracking

Luyuan Gao, Zhihe Zhuang, Hongfeng Tao*, , Yiyang Chen†, Wojciech Paszke‡, Vladimir Stojanovic*§

26.04.2025

Abstract

In practical industries, multiple subsystems are often required to collaborate on a particular repetitive collaborative tracking task, taking a lot of computation. Norm optimal iterative learning control (NOILC) can effectively improve the tracking accuracy for such tasks, and also provide the monotonic convergence of tracking error. However, the high-dimensional matrices and supervectors generated by the lifting technique lead to a computationally expensive problem in the lifted NOILC approach, making it difficult to apply to the collaborative tracking task with long trial length. In order to achieve efficient computation, this paper proposes a novel non-lifted NOILC (N-NOILC) approach for collaborative tracking, with only linear computational complexity regarding the trial length. Exploiting the decomposability of the designed performance criterion, the N-NOILC optimization problem is reformulated as a "sharing" problem, and the alternating direction method of multipliers (ADMM) is introduced for its decentralized solution. Theoretical analysis shows that the proposed algorithm makes the error converge monotonically to zero under the corresponding

*Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Jiangnan University, Wuxi, China, CONTACT Hongfeng Tao. Email: taohongfeng@jiangnan.edu.cn

†School of Mechanical and Electrical Engineering, Soochow University, Suzhou, China

‡Institute of Automation, Electronic and Electrical Engineering, University of Zielona Gora, Zielona Gora, Poland

§Faculty of Mechanical and Civil Engineering, Department of Automatic Control, Robotics and Fluid Technique, University of Kragujevac, 36000 Kraljevo, Serbia, CONTACT Vladimir Stojanovic. Email: vladostojanovic@mts.rs

convergence conditions. Its relevant parameter tuning guidelines are also provided. Finally, the effectiveness of the proposed decentralized N-NOILC approach is verified by numerical simulation.

Keywords: iterative learning control, optimization based learning design, collaborative tracking, computational efficiency

1 Introduction

Collaborative tracking requires multiple subsystems to "join forces" to perform a global objective task, where the global output of the system is regarded as a combination of all subsystem outputs. In nature, it is a very common class of scenarios, e.g., multiple ants carrying food together, centipedes moving on multiple legs, and etc. Such collaborative tracking control for a common output objective also appears in many practical applications, for example: multi-UAV collaborative transportation [14, 31], multiple robotic arms grasp an object together [15], and a task of human-robot object transportation [23], etc. Collaborative tracking systems usually have a sharing center node for collecting and sending information to subsystems so that subsystems can better collaborate with each other to accomplish tasks.

When the task needs to be performed repeatedly, there is a strong motivation to use iterative learning control (ILC) [16,28] to achieve highly accurate collaborative tracking. ILC is an intelligent learning control method for repetitive tasks over a finite time interval. A single execution of the task is called a trial, and the number of sampling points N during a trial is called the trial length. The mechanism of ILC is to continuously improve the input of the current trial by learning the input and output behavior of the previous trial, thus achieving perfect tracking of the desired trajectory.

In recent years, some ILC approaches have been applied to the collaborative tracking problem. Reference 7 introduces ILC to the collaborative tracking scenario for the first time, proposing an inverse-based ILC approach with time-partitioned update that allows to design the input update law for each subsystem individually. For the human-machine collaborative output problem, the inverse-based ILC approach is applied in 21 to improve the system output, which achieves asymptotic convergence to zero tracking error. However, the inverse-based ILC designs lack robustness to model uncertainty. In addition, these approaches do not provide optimal solutions, i.e., there is no performance criterion given to select the optimal input.

The norm optimal ILC (NOILC) [18, 32, 33] approach effectively addresses the above limitations, achieving a variety of compelling collaborative tracking performances by minimizing a specific performance criterion. With adding an input

increment term, NOILC outperforms inverse-based ILC in terms of stability and robustness. Moreover, it has the advantage of ensuring the error monotonic convergence, so the transient problem before error convergence which exists, for example, in 21, is avoided. A generalized decentralized ILC framework along with the corresponding convergence and robustness analyses are given in 5, and a NOILC algorithm is applied to illustrate the effectiveness of the proposed framework. Reference 4 gives a centralized NOILC algorithm for the constrained collaborative tracking problem and further extends a decentralized NOILC implementation.

In terms of the system total computation, compared to single-system trajectory tracking, collaborative tracking tasks often require more computation due to the frequent communication between the network center node and the subsystems. Therefore, a computationally efficient algorithm is attractive. However, the existing NOILC algorithms suffer from a computational efficiency problem that affects their practical applications. The NOILC approach uses the lifting technique for system description that combines information from all sampling time points in each trial into supervectors, resulting in the dimensionality of the vectors and matrices involved are multiples of the trial length N, so the computational complexity of the algorithm grows cubically about N. For some systems that need to perform large tasks, such as industrial robotic systems [12], the number of elements in their system matrices and vectors can reach millions, thus occupying a large amount of memery space and computation time, which is obviously difficult to perform practical computations. Therefore, solving the computational problem is a topic of great practical interest.

For the single-system, several studies have been devoted to solving the above computational problem in NOILC. Reference 11 achieves linear computational complexity by utilizing a sequentially semi-separable structure of lifted system matrices, and reduces the conservatism of the robust design by calculating uncertainty bounds. Reference 13 directly uses the relevant state space equations to solve the optimization problem of the lifted system, obtaining a computationally efficient algorithm. Reference 30 uses a Riccati-based method to solve the optimization problem of lifted system performance criteria, and reveals the relationship between the method and stable inverse. In addition, frequency domain NOILC methods [9, 25] have also been shown to reduce computational complexity compared to lifted NOILC. The schemes designed in the frequency domain replace high dimensional matrix operations in the time domain through filter design, and utilize the fast Fourier transform (FFT) and the independence of frequency domain operations to significantly reduce the computational burden. It is worth noting that the above solutions are realized on the premise that the lifted system is already given, in other words, their work focuses on reducing the impact of choosing the lifted system framework. Fundamentally, the computational com-

plexity is caused by the high dimensional matrix and vector operations generated by the lifting technique. If lifting techniques were not enabled, it would not be a source of computational complexity. Reference 26 takes a completely new strategy, which directly designs the performance criteria of the controlled system with the time index, avoids the use of lifting technique, and develops a computationally efficient non-lifted NOILC (N-NOILC) approach. This approach provided insights for our subsequent study.

The main research goal of this paper is to effectively reduce the computational complexity of the lifted NOILC approach while inheriting its excellent control performance. For the collaborative tracking problem, this greatly reduces the computational time and memory space usage, thus providing more flexibility in the choice of trial length. Inspired by the findings of literature [26], this paper extends the N-NOILC approach to collaborative tracking scenarios, providing a linear computational complexity algorithm.

Moreover, we note that the alternating direction method of multipliers (ADMM) [2] is a very suitable solution method for decentralized convex optimization problems, exhibiting excellent convergence properties under mild conditions. In contrast to the harsh convergence conditions of many decentralized alternative techniques, such as the dual decomposition method [20], which requires strict convexity of the objective function and carefully tuned step sizes. ADMM ensures convergence under milder conditions by leveraging an augmented Lagrangian framework. Moreover, ADMM further distinguishes itself through its ability to decompose complex constraints into tractable subproblems. By introducing auxiliary variables and equality constraints, it decouples intertwined optimization tasks into sequences of simpler operations. Each node independently solves subproblems involving local data or variables, followed by lightweight coordination steps that update global dual variables. Such a design minimizes communication overhead, making ADMM particularly scalable for high-dimensional problems or large datasets partitioned across decentralized systems. These attract us to introduce ADMM to solve the collaborative tracking optimization problem in N-NOILC. Therefore, this paper consider using ADMM design algorithm, proposing a decentralized N-NOILC implementation. The main contributions and highlights of this paper are summarized as follows:

- A computationally efficient N-NOILC algorithm for collaborative tracking is designed. Compared with the existing lifted NOILC algorithms [4, 5], the computational complexity is significantly reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(N)$.

- For the specific collaborative tracking task, we introduce the decentralized ADMM to design an algorithmic framework for parallel computation, which improves the system's collaborative capability and the algorithm's execu-

tion efficiency. A potential advantage is that the global optimal tracking performance of the system is still maintained.

- The relevant properties analysis of the algorithm is given, including convergence, steady state error and computational complexity. The results show that the proposed algorithm allows the collaborative tracking error to converge to zero under very mild parameterization conditions. Further, the corresponding parameter tuning guidelines are provided.

This paper is organized as follows: Section 2 provides the problem formulation and research motivation. A computationally efficient decentralized N-NOILC algorithm based on the ADMM is proposed for collaborative tracking in Section 3. The main results of the proposed algorithm are given in Section 4. Section 5 shows the simulation verification. Conclusions and future work are presented in Section 6.

Notation: Denote the set of natural numbers as $\mathbb{N}$, and the set of positive integers as $\mathbb{N}_+$. $\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ represent the sets of n-dimensional real vectors and $n \times m$ real matrices, respectively. I_n denotes a $n \times n$ identity matrix. $A \succ 0$ represents that the matrix A is positive definite. $\lambda(\cdot)$ is the characteristic value of a matrix. $\rho(\cdot)$ is the spectral radius of a matrix. $\|\cdot\|$ is the 2-norm of a vector or matrix. $\underline{\sigma}(\cdot)$ is the smallest singular value of a matrix. The induced norm of the vector x is defined as $\|x\|_R^2 = x^{\mathrm{T}}Rx$. $X \otimes Y$ represents the Kronecker product of matrices X and Y.

2 Problem Formulation

In this section, the system dynamics and corresponding control objectives are first given. The computational challenges in the lifted NOILC approach are then analyzed, which reveals the main study motivation of this paper.

2.1 System Dynamics

Consider a class of (homogeneous or heterogeneous) networked dynamical systems that consist of s subsystems labeled 1 through s, federating all subsystems to perform a repetitive task collaboratively within a finite time interval. The system dynamics of subsystem i ($1 \le i \le s$) is represented as the following discrete-time linear time-invariant (LTI) model:

$$\begin{aligned} x_{i,k}(t+1) &= A_i x_{i,k}(t) + B_i u_{i,k}(t), \\ y_{i,k}(t) &= C_i x_{i,k}(t). \end{aligned} \tag{1}$$

The discrete-time index is $t = 0, 1, \ldots, N$, the trial index is $k \in \mathbb{N}$, and the trial length is $N \in \mathbb{N}_+$. $x_{i,k}(t) \in \mathbb{R}^n$, $u_{i,k}(t) \in \mathbb{R}^m$ and $y_{i,k}(t) \in \mathbb{R}^l$ are state, input and output variables of subsystem i on trial k, respectively. $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m}$ and $C_i \in \mathbb{R}^{l\times n}$ are system matrices with $l \leq m$.

For the collaborative tracking problem, the global output of the system is the sum of the outputs of all subsystems, i.e.

$$y_k(t) = \sum_{i=1}^{s} y_{i,k}(t) \in \mathbb{R}^l. \tag{2}$$

Further, the global system model is represented as

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t), \\ y_k(t) &= Cx_k(t), \end{aligned} \tag{3}$$

where $u_k(t) = \left[u_{1,k}^{\mathrm{T}}(t), \ldots, u_{s,k}^{\mathrm{T}}(t)\right]^{\mathrm{T}} \in \mathbb{R}^{ms}$, $x_k(t) = \left[x_{1,k}^{\mathrm{T}}(t), \ldots, x_{s,k}^{\mathrm{T}}(t)\right]^{\mathrm{T}} \in \mathbb{R}^{ns}$, $A = \mathrm{diag}\{A_1, \ldots, A_s\} \in \mathbb{R}^{ns\times ns}$, $B = \mathrm{diag}\{B_1, \ldots, B_s\} \in \mathbb{R}^{ns\times ms}$, $C = [C_1, \ldots, C_s] \in \mathbb{R}^{l\times ns}$. The objective of ILC design problem for collaborative tracking is to find a set of appropriate individual control inputs $u_{i,k}(t)$ that allow the system global output $y_k(t)$ track the desired trajectory $y_d(t)$ exactly, and make the tracking error $e_k(t)$ tends to 0 when $k \to \infty$, i.e.

$$\lim_{k\to\infty} e_k(t) = y_d(t) - \lim_{k\to\infty} y_k(t) = 0. \tag{4}$$

Remark 1. *In this paper, the global output is simply defined as the sum of the outputs of all subsystems, as shown in (2). It is a more idealized definition, and similar forms of definitions have been adopted in Literature 4, 7, 21. Literature 5 gives a more general definition of global output as follows:*

$$y_k(t) = \sum_{i=1}^{s} \omega_i y_{i,k}(t),$$

where $\omega_i \in (0, 1]$ is a scalar weight. The weighted sum definition is closer to the realistic scenario where subsystems have different importance and uneven contributions. If the global output is redefined in the form of a weighted sum of the outputs of all subsystems, the collaborative tracking error then becomes

$$e_k(t+1) = y_d(t+1) - \omega CAx_k(t) - \omega CBu_k(t),$$

where $C = \mathrm{diag}\{C_1, \ldots, C_s\} \in \mathbb{R}^{ls\times ms}$, $\omega = [\omega_1, \ldots, \omega_s] \otimes I_l \in \mathbb{R}^{l\times ls}$. By minimizing the collaborative tracking error, the weight matrix ω is introduced into the N-NOILC update law, which affects the convergence performance of the proposed algorithm.

To facilitate the analysis of algorithm properties in the later section, a few reasonable assumptions about the system are given.

Assumption 1. *For all subsystems, each trial satisfies the following state initialization condition:*

$$x_{i,k}(0) = x_{i,d}(0), \tag{5}$$

where $x_{i,k}(0)$ and $x_{i,d}(0)$ are the true and desired state initial values of subsystem i on trial k, respectively.

Assumption 2. *There exists a desired control input $u_d(t)$ that allows the system (3) to achieve perfect tracking of the desired state and output within the finite time interval $[0, N]$, i.e.*

$$\begin{aligned} x_d(t+1) &= Ax_d(t) + Bu_d(t), \\ y_d(t) &= Cx_d(t). \end{aligned} \tag{6}$$

Assumption 3. *System matrix C_iB_i is full row-rank, and the state $x_{i,k}(t)$ is measurable and available.*

Assumption 1 ensures that the state initial value of the system is consistent from trial to trial. Assumption 2 guarantees that the tracking error of the N-NOILC system is possible to converge to 0, i.e., as shown in Eq. (4). Assumption 3 ensures that the system is controllable and the state information is accessible.

2.2 Computational Challenges in Lifted NOILC

As described in the Introduction, the lifted NOILC approach has been decentralized to achieve collaborative tracking of multiple subsystems. Compared to the non-optimization class of algorithms, the lifted NOILC algorithms proposed in [4,5] have the benefit of guaranteeing monotonic convergence of the error along the trial axis, but they cause a significant increase in computational complexity due to the use of lifting technique, thus limiting their practical application. For ease of analysis, the classical lifted NOILC approach for single-system is first briefly described, which is derived by minimizing the following performance criterion:

$$J_{k+1}(\mathbf{u}_{k+1}) = \|\mathbf{e}_{k+1}\|_{\mathbf{Q}}^2 + \|\mathbf{u}_{k+1} - \mathbf{u}_k\|_{\mathbf{R}}^2 . \tag{7}$$

The optimal solution is found as follows:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{L}\mathbf{e}_k, \tag{8}$$

where

$$\mathbf{L} = \left(\mathbf{G}^{\mathrm{T}}\mathbf{Q}\mathbf{G} + \mathbf{R}\right)^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{Q}. \tag{9}$$

In control law (8), the system input $\mathbf{u}_k = \left[u_k^{\mathrm{T}}(0), \ldots, u_k^{\mathrm{T}}(N-1)\right]^{\mathrm{T}}$ is a so-called "supervector", and the system error $\mathbf{e}_k$ is defined accordingly. The lifted system input/output matrix $\mathbf{G}$ in Eq. (9) is defined as

$$\mathbf{G} = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix}.$$

From a computational complexity perspective, the implementation of control law (8) is challenging. The computational complexity is defined as follows [30]:

$$\Delta_{\mathrm{tot}} = \Delta_{\mathrm{init}} + \Delta_{\mathrm{trial}} \cdot k_{\max}, \tag{10}$$

where Δ_{tot} is the total computations of all trials, Δ_{init} is the priori initialization computations, Δ_{trial} is one trial update computations, and $k_{\max}$ is the maximum number of trials.

Since matrix $\mathbf{L}$ is trial-invariant, it can be initialized in advance. Eq. (9) contains three lifted matrix operations: multiplication, addition and inversion. Matrix multiplication has the highest computational complexity of $\mathcal{O}(N^3)$, hence $\Delta_{\mathrm{init}}^{\mathrm{lif}} \sim \mathcal{O}(N^3)$. Eq. (8) shows one trial update computations, containing matrix-vector multiplication and vector-vector addition. Matrix-vector multiplication has the highest computational complexity of $\mathcal{O}(N^2)$, hence $\Delta_{\mathrm{trial}}^{\mathrm{lif}} \sim \mathcal{O}(N^2)$. Therefore, the total computations $\Delta_{\mathrm{tot}}^{\mathrm{lif}} \sim \mathcal{O}(N^3)$. These conclusions are experimentally demonstrated in Section 5.

Essentially, the source of the computational complexity $\mathcal{O}(N^3)$ is the use of $N \times N$ dimensional lifted matrices for system description. Inspired by the research results in [26], this paper proposes a decentralized N-NOILC appproach with efficient computation to solve the collaborative tracking problem.

3 Decentralized N-NOILC Implementation

This section briefly introduces the "sharing" problem in ADMM and provides a decentralized N-NOILC algorithm implementation by transforming the optimization problem of N-NOILC into the "sharing" problem.

3.1 Overview of the "Sharing" Problem in ADMM

ADMM is a powerful optimization method that combines the decomposability of dual ascent with the excellent convergence properties of the method of multipliers. The "sharing" problem is a classical decentralized optimization problem in ADMM.

Many optimization problems can be put in this form, getting the decentralized optimal solutions.

The "sharing" problem is expressed in the following form:

$$\begin{aligned} &\min_{u} \ \sum_{i=1}^{s} f_i(u_i) + g\left(\sum_{i=1}^{s} z_i\right), \\ &\text{s.j.} \ \ u_i - z_i = 0, \quad i = 1, \ldots, s, \end{aligned} \tag{11}$$

where $f_i(\cdot)$ is the local cost function for subsystem i, u_i is the local variable, $g(\cdot)$ is the global cost function for some shared objective of all subsystems, z_i is the auxiliary variable. The "sharing" problem aims to make each subsystem i adjust its local variable u_i to minimize its respective local cost function $f_i(u_i)$ and the shared objective function $g\left(\sum_{i=1}^{s} z_i\right)$.

For the "sharing" problem (11), an iterative solution is performed using ADMM. In the ϑ^{th} iteration, the variables are updated according to the following algorithm:

$$u_i^{\vartheta+1} = \arg\min_{u_i} \left(f_i(u_i) + \frac{\rho}{2} \left\| u_i - u_i^{\vartheta} + \overline{u^{\vartheta}} - \bar{z}^{\vartheta} + \gamma^{\vartheta} \right\|^2 \right), \tag{12}$$

$$\bar{z}^{\vartheta+1} = \arg\min_{\bar{z}} \left(g(s\bar{z}) + \frac{\rho}{2} s \left\| \overline{u^{\vartheta+1}} - \bar{z} + \gamma^{\vartheta} \right\|^2 \right), \tag{13}$$

$$\gamma^{\vartheta+1} = \gamma^{\vartheta} + \overline{u^{\vartheta+1}} - \bar{z}^{\vartheta+1}, \tag{14}$$

where $\overline{u^{\vartheta}} = \frac{1}{s} \sum_{i=1}^{s} u_i^{\vartheta}$, ϑ is the ADMM iteration number, ρ is the penalty parameter, γ is the dual variable. $\bar{z}$ is an additional variable after the simplification operation, transformed from constraint $\bar{z} = \frac{1}{s} \sum_{i=1}^{s} z_i$. The purpose of this is to replace the s components z_i with $\bar{z}$, thus reducing the computational burden on the central node. The specific derivations involved can be found in [2].

The u_i-update (12) is parallel, for $i = 1, \ldots, s$, meaning that each subsystem can perform parallel computations; the $\bar{z}$-update (13) requires the average of all local input variables, and then solves only one optimization problem with respect to $\bar{z}$; the γ-update (14) makes a difference of the results of the first two steps and decentralizes them to each subsystem.

3.2 Reformulated as the "Sharing" Problem

Next, we will build the collaborative tracking problem in the N-NOILC framework and translate it equivalently into a the "sharing" problem.

For lifted NOILC, its high computational cost is attributed to the high-dimensional matrices and vectors generated by the lifting technique. For this reason, we avoid enabling lifting techniques to eliminate the introduction of high-dimensional matrices and vectors at the source. The essence of the N-NOILC approach is to minimize a specific performance criterion that does not involve lifted system. For the global system (3), a centralized performance criterion with respect to $u_{k+1}(t)$ is designed as follows:

$$J_{k+1}\left(u_{k+1}(t)\right) = \|e_{k+1}(t+1)\|_Q^2 + \|\Delta u_{k+1}(t)\|_{\mathcal{R}}^2 , \tag{15}$$

where $\Delta u_{k+1}(t) = u_{k+1}(t) - u_k(t)$ is the input increment along the trial axis designed to ensure smoothness of the input signal. $(Q, \mathcal{R}) \triangleq (qI_l, R \otimes I_s)$ are all symmetric positive definite weighting matrices designed to adjust the weight of tracking error and input increment, where $R = rI_m$. The constructed quadratic performance criterion (15) is improved based on the lifted NOILC. By introducing model information to obtain optimal control gains, it significantly enhances system performance and stability. Therefore, the proposed N-NOILC requires all the information of system matrices A, B, and C, and is essentially a model-based ILC approach.

Remark 2. *Unlike the lifted form in 4, 5, the performance criterion (15) is non-lifted and only contains the information at a sampling time in a trial. It is directly designed for system (3), avoiding using lifted matrices for system description. Therefore, the algorithm for solving it is computationally efficient.*

In the N-NOILC framework, the collaborative tracking problem is formulated as solving the following convex optimization problem:

$$\min_{u} \quad J_{k+1}\left(u_{k+1}(t)\right) = \|e_{k+1}(t+1)\|_Q^2 + \|\Delta u_{k+1}(t)\|_{\mathcal{R}}^2 . \tag{16}$$

The collaborative tracking problem (16) can be viewed as a system global minimization problem since the performance criterion encompasses all the subsystems. That is, it is a control problem for a centralized architecture.

By reviewing the "sharing" problem in the previous subsection, we can see that both are essentially the same, they are minimization problems for a specific objective function. As long as the centralized performance criterion $J_{k+1}\left(u_{k+1}(t)\right)$ are decomposable, we have a strong motivation to transform the collaborative tracking problem on N-NOILC into a "sharing" problem and introduce decentralized ADMM for its solution. For the penalty term $\|\Delta u_{k+1}(t)\|_{\mathcal{R}}^2$ in the performance criterion, it is decomposed into the form of an accumulation of all subsystem input increments. For the error term $\|e_{k+1}(t+1)\|_Q^2$, although it cannot be decomposed into a number of subterms that add up, it is regarded as a global shared objective

of the system, equivalent to $g\left(\sum_{i=1}^{s} z_i\right)$ in problem (11). In this way, the collaborative tracking problem (16) is equivalently transformed into the following "sharing" problem:

$$\begin{aligned} \min_{u} \quad & \sum_{i=1}^{s} f_{i,k+1}\left(u_{i,k+1}(t)\right) + g_{k+1}\left(\sum_{i=1}^{s} z_{i,k+1}(t)\right), \\ \text{s.j.} \quad & C_i B_i u_{i,k+1}(t) - z_{i,k+1}(t) = 0, \quad i = 1, \ldots, s. \end{aligned} \tag{17}$$

The local cost function and the global shared objective function are respectively expressed as follows:

$$f_{i,k+1}\left(u_{i,k+1}(t)\right) = \left\|u_{i,k+1}(t) - u_{i,k}(t)\right\|_R^2, \tag{18}$$

$$g_{k+1}\left(\sum_{i=1}^{s} z_{i,k+1}(t)\right) = \left\| \; e_k(t+1) - CA\Delta x_{k+1}(t) + CBu_k(t) - \sum_{i=1}^{s} z_{i,k+1}(t) \; \right\|_Q^2. \tag{19}$$

The local cost function $f_{i,k+1}\left(u_{i,k+1}(t)\right)$ represents the input increment of subsystem i. It enables a smooth transition of control inputs between trials by penalizing the changes of control inputs in neighboring trials, which helps to improve the stability and robustness of the system throughout the iterative learning process. This design of the local cost function allows the overall optimization problem to be decomposed into local optimization problems for individual subsystems, thus providing a prerequisite for parallel solution using sharing ADMM. The global shared objective function $g_{k+1}\left(\sum_{i=1}^{s} z_{i,k+1}(t)\right)$ is designed to minimize the collaborative tracking error of the current trial, despite the fact that the error term is indecomposable. By incorporating the variables $z_{i,k+1}(t)$ associated with the subsystems and some system global shared terms, the function satisfies the constraint relationship between the different variables, taking the error of the current trial as the system shared objective. To illustrate equivalence, Remark 3 is provided.

Remark 3. *The first term in the "sharing" problem (17) is equal to the second term in the performance criterion (15), i.e.*

$$\sum_{i=1}^{s} f_{i,k+1}\left(u_{i,k+1}(t)\right) = \left\|\Delta u_{k+1}(t)\right\|_{\mathcal{R}}^2. \tag{20}$$

Substituting the constraint condition $C_i B_i u_{i,k+1}(t) = z_{i,k+1}(t)$ into the second term in the "sharing" problem (17) yields the first term in the performance criterion (15), i.e.

$$g_{k+1}\left(\sum_{i=1}^{s} z_{i,k+1}(t)\right) = \left\|e_{k+1}(t+1)\right\|_Q^2. \tag{21}$$

Therefore, solving the "sharing" problem (17) using ADMM is equivalent to solving the collaborative tracking problem (16) in a decentralized manner.

3.3 Decentralized N-NOILC Algorithm

Obviously, the optimization problem (17) is a "sharing" problem incorporating the N-NOILC implementation framework. The ADMM is employed to provide a decentralized solution to the "sharing" problem, which can hence develop a decentralized N-NOILC algorithm for the collaborative tracking problem (16). Similar to the algorithm (12)-(14), the following iteration algorithm is yielded:

$$u_{i,k+1}^{\vartheta+1}(t) = \underset{u_{i,k+1}(t)}{\arg\min} \left(\begin{array}{l} \|u_{i,k+1}(t) - u_{i,k}(t)\|_R^2 \\ + \dfrac{\rho}{2} \left\| C_i B_i u_{i,k+1}(t) - C_i B_i u_{i,k+1}^{\vartheta}(t) \right. \\ \qquad \left. + \overline{CBu_{k+1}^{\vartheta}(t)} - \bar{z}_{k+1}^{\vartheta}(t) + \gamma_{k+1}^{\vartheta}(t) \right\|^2 \end{array} \right), \tag{22}$$

$$\bar{z}_{k+1}^{\vartheta+1}(t) = \underset{\bar{z}_{k+1}(t)}{\arg\min} \left(\begin{array}{l} \left\| e_k(t+1) - CA\Delta x_{k+1}(t) \right. \\ \qquad \left. + CBu_k(t) - s\bar{z}_{k+1}(t) \right\|_Q^2 \\ + \dfrac{\rho}{2} s \left\| \overline{CBu_{k+1}^{\vartheta+1}(t)} - \bar{z}_{k+1}(t) + \gamma_{k+1}^{\vartheta}(t) \right\|^2 \end{array} \right), \tag{23}$$

$$\gamma_{k+1}^{\vartheta+1}(t) = \gamma_{k+1}^{\vartheta}(t) + \overline{CBu_{k+1}^{\vartheta+1}(t)} - \bar{z}_{k+1}^{\vartheta+1}(t), \tag{24}$$

where $\overline{CBu_{k+1}^{\vartheta}(t)} = \frac{1}{s}\sum_{i=1}^{s} C_i B_i u_{i,k+1}^{\vartheta}(t)$, $\bar{z}_{k+1}^{\vartheta}(t)$ is the additional variable due to the simplification operation. Further, the analytic expression of the algorithm (22)-(24) is obtained as follows:

$$u_{i,k+1}^{\vartheta+1}(t) = \left(2R + \rho(C_iB_i)^{\mathrm{T}}(C_iB_i)\right)^{-1} \left(\begin{array}{l} 2R\, u_{i,k}(t) + \rho(C_iB_i)^{\mathrm{T}} \\ \cdot \left(C_iB_iu_{i,k+1}^{\vartheta}(t) - \overline{CBu_{k+1}^{\vartheta}(t)} \right. \\ \qquad \left. + \bar{z}_{k+1}^{\vartheta}(t) - \gamma_{k+1}^{\vartheta}(t) \right) \end{array} \right), \tag{25}$$

$$\bar{z}_{k+1}^{\vartheta+1}(t) = \left(sQ + \frac{\rho}{2} I_l\right)^{-1} \left(\begin{array}{l} Q\left(e_k(t+1) - CA\Delta x_{k+1}(t) + CB\, u_k(t)\right) \\ + \dfrac{\rho}{2}\left(\overline{CBu_{k+1}^{\vartheta+1}(t)} + \gamma_{k+1}^{\vartheta}(t)\right) \end{array} \right), \tag{26}$$

$$\gamma_{k+1}^{\vartheta+1}(t) = \gamma_{k+1}^{\vartheta}(t) + \overline{CBu_{k+1}^{\vartheta+1}(t)} - \bar{z}_{k+1}^{\vartheta+1}(t). \tag{27}$$

The first step (25) represents all the local variable updates, where each subsystem gets its respective optimal control input in a parallel computation. The

second step (26) indicates additional variable update that require system global information sharing. The third step (27) represents the update of the dual variable, which is realized by making a difference between the results of the first two steps. When updating one of the steps, the other two steps are fixed, and thus all variables are updated alternately. Repeat the above three steps until all variables no longer change, achieving convergence ultimately.

To clearly illustrate the execution of the algorithm, the corresponding pseudocode is shown in Algorithm 1.

Algorithm 1 Decentralized N-NOILC algorithm for collaborative tracking

Input: System matrices A_i, B_i and C_i; desired trajectory $y_d(t)$; weighting matrices Q and R; the trial length N; maximum trial index $k_{\max}$, maximum ADMM iteration index $\vartheta_{\max}$, positive penalty parameter ρ

Output: Optimal control input $u_{i,k}(t)$ for each subsystem i

Initialization: Set initial state value $x_{i,k}(0) = x_0$
for k=0 to $k_{\max}$ **do**
    for t=0 to N **do**
        for ϑ=0 to $\vartheta_{\max}$ **do**
            Each subsystem updates its inputs in parallel by (25)
            All subsystems send inputs to the system center node
            The system center node updates variables by (26), (27)
            The system center node sends results to all subsystems
        end for
        Assign the value of $u_{i,k+1}^{\vartheta_{\max}}(t)$ to $u_{i,k+1}(t)$
    end for
Return: Optimal control input $u_{i,k+1}(t)$
end for

Remark 4. *The proposed decentralized design solves the collaborative tracking problem by introducing the shared ADMM, unlike many existing distributed ILC schemes, whether model-based [8] or data-driven [3]. Compared with the proposed decentralized design, the most significant difference of distributed design lies in that the update of subsystem local variables needs to rely on knowledge from their surrounding neighbors. This inherent control architecture naturally leads them to address consensus tracking problems. In addition, benefiting from the characteristics of shared ADMM, the proposed design may have lower computational complexity and communication burden. The update of local variables in shared ADMM only involves information from local nodes, eliminating the need for information communication and interaction with neighboring nodes. As well, each update of global variables and dual variables only needs to be calculated once at the central*

node, avoiding parallel computation of s components and thus reducing computational complexity. In contrast, the distributed approach, such as in 8, requires more computational and communication overhead.

4 Main Results

This section provides various features of Algorithm 1, including convergence property, parameter effects and computational complexity.

4.1 Convergence Analysis

Our motivation for introducing ADMM is to achieve a decentralized solution to the collaborative tracking problem, which leads to a different execution of Algorithm 1 than the regular two-dimensional ILC algorithms. For each discrete-time point $t \in [0, N]$, the inputs $u^{\vartheta}_{i,k+1}(t)$ need to go through $\vartheta_{\max}$ ADMM iterations before converging to the optimal solution. Therefore, the convergence analysis is divided into two parts: first the convergence of the input variables updated according to the ADMM iterations is analyzed, and then the convergence of their convergence results along the trial axis is discussed.

For ADMM, the following proposition guarantee convergence [2]:

Proposition 1. *When the penalty parameter $\rho > 0$, applying Algorithm 1 will achieve the following convergence properties:*

- *Residual convergence:* $\lim_{\vartheta\to\infty} \left(C_i B_i u^{\vartheta}_{i,k+1}(t) - z^{\vartheta}_{i,k+1}(t)\right) = 0$, *implying the feasibility of iterations.*
- *Cost function convergence:* $\sum_{i=1}^{s} f_{i,k+1}\left(u^{\vartheta}_{i,k+1}(t)\right) + g_{k+1}\left(\sum_{i=1}^{s} z^{\vartheta}_{i,k+1}(t)\right)$ *approaches the optimal value as $\vartheta \to \infty$.*

With Proposition 1, the input convergence result obtained by applying Algorithm 1 is given next.

Theorem 4.1. *Let the proposed Algorithm 1 be applied. When ADMM iteration number $\vartheta \to \infty$, the iteratively updated inputs of each subsystem converge to the unique optimal solution of the centralized collaborative tracking problem (16), i.e.*

$$\lim_{\vartheta\to\infty} \left({u^{\vartheta}_{1,k+1}}^{\mathrm{T}}(t), \ldots, {u^{\vartheta}_{s,k+1}}^{\mathrm{T}}(t)\right)^{\mathrm{T}} = \arg\min_{u_{k+1}(t)} \left\{J\left(u_{k+1}(t)\right)\right\}. \tag{28}$$

Proof. Substituting to the error definition, the performance criterion (15) is written as the following quadratic equation:

$$J_{k+1}\left(u_{k+1}(t)\right) = u_{k+1}^{\mathrm{T}}(t) M u_{k+1}(t) + 2b^{\mathrm{T}} u_{k+1}(t) + c, \tag{29}$$

where

$$\begin{aligned} M &= (CB)^{\mathrm{T}} QCB + \mathcal{R}, \\ W(t) &= r(t+1) - CAx_{k+1}(t), \\ b^{\mathrm{T}} &= -\left(W^{\mathrm{T}}(t) QCB + u_k^{\mathrm{T}}(t)\mathcal{R}\right), \\ c &= u_k^{\mathrm{T}}(t)\mathcal{R}u_k(t) + W^{\mathrm{T}}(t) Q W(t). \end{aligned}$$

According to Assumption 3, M is a positive definite matrix since CB is full row-rank and Q, $\mathcal{R}$ are all positive definite, making $J_{k+1}(u_{k+1}(t))$ a convex function. Thus, there exists a unique optimal input minimizing $J_{k+1}(u_{k+1}(t))$ at each discrete-time point $t \in [0, N]$. Also, the cost function in problem (17) is a convex function and the corresponding constraints are convex sets, so there also exists a unique optimal solution to problem (17).

According to Proposition 1, when $\vartheta \to \infty$, the input convergence value of each subsystem minimizes the cost function in (17). Furthermore, the constraint $C_i B_i u_{i,k+1}(t) - z_{i,k+1}(t) = 0$, for all $i = 1, \ldots, s$, will be satisfied. In this case, as described in Remark 3, the cost function in (17) is equivalent to the performance criterion (15). Therefore, for all discrete-time point $t \in [0, N]$ during one trial, the input convergence value of each subsystem obtained by applying Algorithm 1 is the same as the optimal solution of the collaborative tracking problem (16), i.e

$$\lim_{\vartheta \to \infty} \left({u_{1,k+1}^{\vartheta}}^{\mathrm{T}}(t), \ldots, {u_{s,k+1}^{\vartheta}}^{\mathrm{T}}(t) \right)^{\mathrm{T}} = \arg\min_{u_{k+1}(t)} \left\{ J\left(u_{k+1}(t)\right) \right\}.$$

Theorem 4.1 is proved. □

Theorem 4.1 indicates that all input variables eventually converge to the optimal solution of the collaborative tracking problem (16) after ADMM iterations.

Conventional time-domain control is concerned with what happens to the system as time tends to infinity. ILC is more concerned with what happens to the system after numerous trials. In ILC terminology, "system stability" and "convergence" are often used interchangeably. For a more rigorous description, we next give a relevant definition of the stability of an ILC system, and the conditions, i.e., the convergence conditions, that are used to test whether a given system is stable in this sense. The definitions and analysis of convergence in this paper are both conducted from the perspective of control inputs.

Definition 4.2. *(Asymptotic Convergence [19]) For an ILC system, if*

$$\lim_{k\to\infty} \|\mathbf{u}_d - \mathbf{u}_k\| = 0, \tag{30}$$

holds , then the ILC system is asymptotically convergent.

According to the Definition 4.2, it can be seen that asymptotic convergence is only concerned with the final convergence result of the control inputs and not with the convergence behavior along the trial axis. Therefore, the asymptotically stable N-NOILC system may experience large transients along the trial axis during convergence. Another form of stability, monotonic convergence, that is "stronger" than asymptotic convergence, which avoids the possibility of transient growth appearing, is described next.

Definition 4.3. *(Monotonic Convergence [29]) For an ILC system, if*

$$\|\mathbf{u}_{k+1} - \mathbf{u}_\infty\| \leq \|\mathbf{u}_k - \mathbf{u}_\infty\|, \tag{31}$$

holds , then the ILC system is monotonically convergent.

Monotonic convergence requires not only that $\lim_{k\to\infty} \mathbf{u}_k = \mathbf{u}_\infty$ exists, but also that the input error of the current trial cannot be larger than that of the previous trial, which is a more stringent convergence condition. Below, the following theorems are deduced to give the convergence analysis of the optimal inputs in the trial domain.

Theorem 4.4. *If the system (1) is controlled by the N-NOILC algorithm 1, then the input variables are asymptotically convergent along the trial axis.*

Proof. Following the convergence result of Theorem 4.1, we further use the optimality condition to yield the optimal solution to the collaborative tracking problem as follows:

$$u_{k+1}(t) = u_k(t) + L_e e_k(t+1) - L_x \Delta x_{k+1}(t). \tag{32}$$

The centralized solution (32) is a column stack vector that can be viewed as the converged value of all subsystems' input after the ADMM iterations, where

$$L_e = \left[(CB)^{\mathrm{T}} QCB + \mathcal{R}\right]^{-1} (CB)^{\mathrm{T}} Q, \tag{33}$$

$$L_x = \left[(CB)^{\mathrm{T}} QCB + \mathcal{R}\right]^{-1} (CB)^{\mathrm{T}} QCA. \tag{34}$$

According to the error definition, has

$$e_k(t+1) = y_d(t+1) - C(Ax_k(t) + Bu_k(t)). \tag{35}$$

Substituting (35) into (32) yields

$$u_{k+1}(t) = L_u u_k(t) - L_x x_{k+1}(t) + L_e y_d(t+1), \tag{36}$$

where

$$L_u = \left[(CB)^{\mathrm{T}} Q C B + \mathcal{R}\right]^{-1} \mathcal{R}. \tag{37}$$

According to the definition of system model and Assumption 1, yields

$$x_{k+1}(t) = A^t x_d(0) + \sum_{v=0}^{t-1} A^{t-1-v} B u_{k+1}(v). \tag{38}$$

Substituting (38) into (36) yields

$$\sum_{v=0}^{t-1} \eta_{t,v} u_{k+1}(v) + u_{k+1}(t) = L_u u_k(t) + \theta(t), \tag{39}$$

where $\theta(t) = L_e y_d(t+1) - L_x A^t x_d(0)$ and $\eta_{t,v} = L_x A^{t-1-v} B$. Then (39) is written in a lifted form as

$$\mathbf{u}_{k+1} = \mathbf{H}^{-1}\mathbf{L}\mathbf{u}_k + \mathbf{H}^{-1}\boldsymbol{\theta}, \tag{40}$$

where

$$\begin{aligned}
\mathbf{u}_{k+1} &= \left[u_{k+1}^{\mathrm{T}}(0), \ldots, u_{k+1}^{\mathrm{T}}(N-1)\right]^{\mathrm{T}} \in \mathbb{R}^{msN}, \\
\mathbf{u}_k &= \left[u_k^{\mathrm{T}}(0), \ldots, u_k^{\mathrm{T}}(N-1)\right]^{\mathrm{T}} \in \mathbb{R}^{msN}, \\
\boldsymbol{\theta} &= \left[\theta^{\mathrm{T}}(0), \ldots, \theta^{\mathrm{T}}(N-1)\right]^{\mathrm{T}} \in \mathbb{R}^{msN}, \\
\mathbf{H} &= \begin{bmatrix} I & 0 & \cdots & 0 \\ \eta_{1,0} & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{N-1,0} & \cdots & \eta_{N-1,N-2} & I \end{bmatrix} \in \mathbb{R}^{msN \times msN}, \\
\mathbf{L} &= \begin{bmatrix} L_u & & 0 \\ & \ddots & \\ 0 & & L_u \end{bmatrix} \in \mathbb{R}^{msN \times msN}.
\end{aligned}$$

For the lifted law (40), the asymptotic convergence condition is $\rho(\mathbf{H}^{-1}\mathbf{L}) < 1$ [28]. The matrix $\mathbf{H}$ is a lower triangular matrix with the unit array as diagonal element, so there is $\lambda(\mathbf{H}^{-1}\mathbf{L}) = \lambda(\mathbf{L})$, i.e., $\rho(\mathbf{H}^{-1}\mathbf{L}) = \rho(L_u)$. In this case, since the matrices Q and $\mathcal{R}$ are all symmetric positive definite matrices, and $\mathcal{R} = rI_{ms}$ with $r > 0$, further yields

$$\rho(L_u) \leq \|L_u\| = \left\| \left[(CB)^{\mathrm{T}} Q C B + \mathcal{R}\right]^{-1} \mathcal{R} \right\| < 1, \tag{41}$$

i.e., $\rho(\mathbf{H}^{-1}\mathbf{L}) < 1$. Theorem 4.4 is proved. □

Remark 5. *According to Theorem 4.4, asymptotic convergence of Algorithm 1 is very easy to satisfy, requiring only that Q and $\mathcal{R}$ are symmetric positive definite matrices, and $\mathcal{R} = rI_{ms}$ with $r > 0$.*

During the convergence of the algorithm, large transient growth [17] should be avoided as much as possible, so monotonic convergence is desirable. To ensure monotonic convergence, the corresponding Theorem 4.5 is given.

Theorem 4.5. *Let the proposed Algorithm 1 be applied. If the symmetric positive definite matrices Q and $\mathcal{R}$ make the following condition hold:*

$$\mathbf{P}^{\mathrm{T}} + \mathbf{P} \succ 0, \tag{42}$$

then the input variables are monotonic convergent along the trial axis, where $\mathbf{P} = (\mathbf{CB})^{\mathrm{T}}\mathbf{QC}(\boldsymbol{\phi} + \mathbf{B})$. Define the following lifted matrices:

$$\boldsymbol{\phi} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ AB & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & \cdots & AB & 0 \end{bmatrix} \in \mathbb{R}^{nsN \times msN},$$

$$\mathbf{B} = \begin{bmatrix} B & & 0 \\ & \ddots & \\ 0 & & B \end{bmatrix} \in \mathbb{R}^{nsN \times msN},$$

$\mathbf{C} \in \mathbb{R}^{lN \times nsN}$, $\mathbf{Q} \in \mathbb{R}^{lN \times lN}$, $\mathbf{R} \in \mathbb{R}^{msN \times msN}$ *are defined accordingly.*

Proof. For the lifted law (40), the monotonic convergence condition is $\left\|\mathbf{H}^{-1}\mathbf{L}\right\| < 1$ [24]. The matrices $\mathbf{H}$ and $\mathbf{L}$ are written as follows:

$$\mathbf{H} = \left[(\mathbf{CB})^{\mathrm{T}}\mathbf{QCB} + \mathbf{R}\right]^{-1}(\mathbf{CB})^{\mathrm{T}}\mathbf{QC}\boldsymbol{\phi} + \mathbf{I}, \tag{43}$$

$$\mathbf{L} = \left[(\mathbf{CB})^{\mathrm{T}}\mathbf{QCB} + \mathbf{R}\right]^{-1}\mathbf{R}. \tag{44}$$

Combining (43) and (44) yields,

$$\begin{aligned} \mathbf{H}^{-1}\mathbf{L} &= \left[(\mathbf{CB})^{\mathrm{T}}\mathbf{QC}(\boldsymbol{\phi} + \mathbf{B}) + \mathbf{R}\right]^{-1}\mathbf{R} \\ &= (\mathbf{P} + \mathbf{R})^{-1}\mathbf{R}. \end{aligned} \tag{45}$$

Taking 2-norm on both sides of the equal sign yields

$$\begin{aligned} \left\|\mathbf{H}^{-1}\mathbf{L}\right\| &\leq \left\|(\mathbf{P} + \mathbf{R})^{-1}\right\| \|\mathbf{R}\| \\ &= \frac{r}{\underline{\sigma}(\mathbf{P} + \mathbf{R})}. \end{aligned} \tag{46}$$

According to the definition of singular value, yields

$$\begin{aligned}\underline{\sigma}\left(\mathbf{P}+\mathbf{R}\right) &= \sqrt{\lambda_{\min}\left(\left(\mathbf{P}+\mathbf{R}\right)^{\mathrm{T}}\left(\mathbf{P}+\mathbf{R}\right)\right)} \\ &= \sqrt{\lambda_{\min}\left(\mathbf{P}^{\mathrm{T}}\mathbf{P}+r(\mathbf{P}^{\mathrm{T}}+\mathbf{P})\right)+r^2}.\end{aligned} \tag{47}$$

Thus , if $\mathbf{P}^{\mathrm{T}}+\mathbf{P}\succ 0$ holds, we get

$$\lambda_{\min}\left(\mathbf{P}^{\mathrm{T}}\mathbf{P}+r(\mathbf{P}^{\mathrm{T}}+\mathbf{P})\right)>0. \tag{48}$$

From this, it ultimately follows that

$$\left\|\mathbf{H}^{-1}\mathbf{L}\right\| \leq \frac{r}{\sqrt{\lambda_{\min}\left(\mathbf{P}^{\mathrm{T}}\mathbf{P}+r(\mathbf{P}^{\mathrm{T}}+\mathbf{P})\right)+r^2}}<1. \tag{49}$$

Theorem 4.5 is proved. □

To sum up, the convergence of Algorithm 1 is guaranteed as long as the above theorems are satisfied by tuning the parameters.

Remark 6. *Although Theorem 4.5 shows that monotone convergence requires some positive definite condition for a certain matrix, it follows from Theorem 4.1 and 4.4 that asymptotic convergence can be guaranteed by fulfilling only some very mild parameterization conditions (requiring Q, $\mathcal{R}$ are symmetric positive definite matrices and $\rho>0$). This implies that the specific values of the relevant parameters do not affect the tendency of the input variables to converge asymptotically. Moreover, the tracking error converges to zero along the trial axis in this case (see Section 4.2). To optimize the convergence speed, the parameters tuning guidelines are given in Section 4.3.*

Remark 7. *Process disturbances in system models are common in engineering practice. With the introduction of stochastic disturbance $d_k(t)$, the global model (3) is represented as follows:*

$$\begin{aligned}\hat{x}_k(t+1) &= A\hat{x}_k(t)+Bu_k(t)+d_k(t), \\ y_k(t) &= C\hat{x}_k(t),\end{aligned} \tag{50}$$

and the collaborative tracking error is rewritten as

$$\hat{e}_k(t+1)=y_d(t+1)-CAx_k(t)-CBu_k(t)-Cd_k(t). \tag{51}$$

A simple analysis is conducted on how disturbances impact the proposed algorithm. Substituting the constraint condition $C_iB_iu_{i,k+1}(t)=z_{i,k+1}(t)$ into the global shared objective function (17), the collaborative tracking problem is reformulated as

$$\min_{u} \quad \hat{J}_{k+1}\left(u_{k+1}(t)\right)=\left\|\hat{e}_{k+1}(t+1)+C\Delta d_{k+1}(t)\right\|_{Q}^{2}+\left\|\Delta u_{k+1}(t)\right\|_{\mathcal{R}}^{2}, \tag{52}$$

where $\Delta d_{k+1}(t) = d_{k+1}(t) - d_k(t)$. *In this case, the global optimal solution (32) to the collaborative tracking problem becomes*

$$u_{k+1}(t) = u_k(t) + L_e\hat{e}_k(t+1) - L_x\Delta\hat{x}_{k+1}(t). \tag{53}$$

Substituting (51) into (53) yields

$$u_{k+1}(t) = L_u u_k(t) - L_x\hat{x}_{k+1}(t) + L_e y_d(t+1) - L_e C d_k(t), \tag{54}$$

According to the definition of system model (50), yields

$$\hat{x}_{k+1}(t) = A^t x_d(0) + \sum_{v=0}^{t-1} A^{t-1-v} B u_{k+1}(v) + \sum_{v=0}^{t-1} A^{t-1-v} d_{k+1}(v). \tag{55}$$

Substituting (55) into (54) yields

$$\sum_{v=0}^{t-1} \eta_{t,v} u_{k+1}(v) + u_{k+1}(t) = L_u u_k(t) + \theta(t) - L_e C d_k(t) - \sum_{v=0}^{t-1} \xi_{t,v} d_{k+1}(v), \tag{56}$$

where $\xi_{t,v} = L_x A^{t-1-v}$. *Then (56) is written in a lifted form as*

$$\mathbf{u}_{k+1} = \mathbf{H}^{-1}\mathbf{L}\mathbf{u}_k + \mathbf{H}^{-1}\boldsymbol{\theta} - \mathbf{H}^{-1}\mathbf{S}\mathbf{d}_k - \mathbf{H}^{-1}\mathbf{K}\mathbf{d}_{k+1}, \tag{57}$$

where

$$\begin{aligned}
\mathbf{d}_{k+1} &= \left[d_{k+1}^{\mathrm{T}}(0), \ldots, d_{k+1}^{\mathrm{T}}(N-1)\right]^{\mathrm{T}} \in \mathbb{R}^{msN}, \\
\mathbf{d}_k &= \left[d_k^{\mathrm{T}}(0), \ldots, d_k^{\mathrm{T}}(N-1)\right]^{\mathrm{T}} \in \mathbb{R}^{msN}, \\
\mathbf{K} &= \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \xi_{1,0} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{N-1,0} & \cdots & \xi_{N-1,N-2} & 0 \end{bmatrix} \in \mathbb{R}^{msN \times msN}, \\
\mathbf{S} &= \begin{bmatrix} L_e C & & 0 \\ & \ddots & \\ 0 & & L_e C \end{bmatrix} \in \mathbb{R}^{msN \times msN}.
\end{aligned}$$

If the disturbance $\mathbf{d}_k$ *is bounded, then the input* $\mathbf{u}_{k+1}$ *is also bounded. Also, the steady-state error does not converge to 0, but converges within an upper bound related to the magnitude of the disturbance.*

4.2 Steady-State Error

For the ILC controlled system, the control performance is usually measured by the steady-state error along the trial axis, with smaller steady-state error implying better control performance. The steady-state error is defined with the following notation:

$$e_{ss}(t) = \lim_{k\to\infty} e_k(t) = e_\infty(t). \tag{58}$$

According to Eq. (36), yields

$$u_\infty(t) = \Big((CB)^{\mathrm{T}} QCB\Big)^{-1} (CB)^{\mathrm{T}} Q\left(y_d(t+1) - CAx_\infty(t)\right). \tag{59}$$

According to the error definition, has

$$e_\infty(t+1) = y_d(t+1) - C(Ax_\infty(t) + Bu_\infty(t)). \tag{60}$$

Substituting (59) into (60) yields

$$e_\infty(t+1) = 0. \tag{61}$$

Therefore, the steady-state error of the system can converge to zero, demonstrating the superior control performance of Algorithm 1.

4.3 Tuning of Relevant Parameters

For Algorithm 1, the penalty parameter ρ and the weighting matrices Q, R have an impact on the convergence performance of the algorithm. Their tuning are discussed separately below.

4.3.1 The penalty parameter ρ

Theoretically, input variables $u^{\vartheta}_{i,k+1}(t)$ converges to the optimal solution when $\vartheta \to \infty$. However, in practice, ϑ does not need to be set particularly large to be sufficient to achieve very good convergence accuracy, as long as the penalty parameter ρ value is appropriate. Although ADMM guarantees convergence for any positive penalty parameter ρ, the value of ρ still affects the convergence speed. If this parameter is not chosen properly, it may make the convergence of ADMM iterations slower. Intuitively, the penalty parameter profoundly affects the convergence speed by determining the penalty strength of the quadratic term. A larger ρ strengthens the quadratic penalty term $\frac{\rho}{2}\|\cdot\|^2$, hastening early-stage feasibility. However, excessive ρ values may induce ill-conditioning in the u- and z-subproblems. Conversely, smaller ρ relaxes the penalty, simplifying subproblem solutions but risking slower convergence due to persistent primal-dual residual

imbalances or oscillatory updates. Some studies have also proposed methods for selecting the parameter ρ, e.g., by following rules of thumb [2,6], or by mathematical calculations [10,22,27]. The above literature has shown that we can establish that global convergence of ADMM iterations at a linear speed by optimizing ρ, provided that the cost function is convex. Next, the explicit expression for the optimal penalty parameter is provided.

In order to characterize the convergence speed of ADMM, the asymptotic convergence factor of the input sequence $\left\{u_{k+1}^{\vartheta}(t)\right\}$ converging to the optimal input $u_{k+1}^{*}(t)$ is defined as follows:

$$\zeta \triangleq \sup_{u_{k+1}^{0}(t)\neq u_{k+1}^{*}(t)} \left(\frac{\left\| u_{k+1}^{\vartheta}(t) - u_{k+1}^{*}(t) \right\|}{\left\| u_{k+1}^{0}(t) - u_{k+1}^{*}(t) \right\|} \right). \tag{62}$$

A smaller convergence factor means that the ADMM iterations converge faster. To find the optimal ρ, we formulate the selection of the optimal parameters as a equation-constrained quadratic programming problem:

$$\begin{aligned} &\min_{u,z} \ u_{k+1}^{\mathrm{T}}(t) M u_{k+1}(t) + 2b^{\mathrm{T}} u_{k+1}(t) + c, \\ &\text{s.j.} \ \ CBu_{k+1}(t) - z_{k+1}(t) = 0, \end{aligned} \tag{63}$$

where

$$\begin{aligned} M &= (CB)^{\mathrm{T}} Q CB + \mathcal{R}, \\ W(t) &= r(t+1) - CAx_{k+1}(t), \\ b^{\mathrm{T}} &= -\left(W^{\mathrm{T}}(t) Q CB + u_{k}^{\mathrm{T}}(t)\mathcal{R} \right), \\ c &= u_{k}^{\mathrm{T}}(t)\mathcal{R}u_{k}(t) + W^{\mathrm{T}}(t) Q W(t). \end{aligned}$$

Based on the results of Literature [10], the following proposition on optimal parameter selection is derived.

Proposition 2. *If the constraint matrix CB is either full row-rank or invertible, then the optimal penalty parameter ρ^* that minimize the convergence factor is:*

$$\rho^{*} = \left(\sqrt{\lambda_{\min}\left(CBM^{-1}(CB)^{\mathrm{T}} \right) \lambda_{\max}\left(CBM^{-1}(CB)^{\mathrm{T}} \right)} \right)^{-1}. \tag{64}$$

Proposition 2 gives an explicit expression for computing the optimal penalty parameter that ensure the smallest convergence factor. In this way, we can accelerate the global linear convergence of ADMM iterations by choosing the penalty parameter, which is beneficial in practice.

4.3.2 The weighting matrix R

According to the performance criterion (15), the weighting matrix R determines the convergence speed of the algorithm along the trial axis (selecting $Q = I_l$). The matrix R adjusts the weights of $\|e_{k+1}(t+1)\|^2$ and $\|\Delta u_{k+1}(t)\|^2$ in the performance criterion. Intuitively, if R is large, the algorithm will focus on optimizing $\|\Delta u_{k+1}(t)\|^2$ so that the inputs are close to each other between adjacent trials; if R is small, the algorithm will focus on achieving the largest possible decay of $\|e_{k+1}(t+1)\|^2$ along the trial axis by adjusting the inputs. The effect of the value of R on the convergence speed is discussed below.

When $R = 0$, it follows from (36) that

$$u_{k+1}(t) = \Big((CB)^{\mathrm{T}} QCB\Big)^{-1} (CB)^{\mathrm{T}} Q\left(y_d(t+1) - CAx_{k+1}(t)\right). \tag{65}$$

According to Assumption 1 and Eq. (65), yields

$$u_k(0) = \Big((CB)^{\mathrm{T}} QCB\Big)^{-1} (CB)^{\mathrm{T}} Q\left(y_d(1) - CAx_d(0)\right), \tag{66}$$

thus

$$\begin{aligned} y_{k+1}(1) &= CAx_d(0) + CBu_k(0) \\ &= CAx_d(0) + y_d(1) - CAx_d(0) \\ &= y_d(1). \end{aligned} \tag{67}$$

According to Assumption 2, yields $x_k(1) = x_d(1)$, thus

$$u_k(1) = \Big((CB)^{\mathrm{T}} QCB\Big)^{-1} (CB)^{\mathrm{T}} Q\left(y_d(2) - CAx_d(1)\right). \tag{68}$$

Further, yields

$$\begin{aligned} y_{k+1}(2) &= CAx_d(1) + CBu_k(1) \\ &= CAx_d(1) + y_d(2) - CAx_d(1) \\ &= y_d(2). \end{aligned} \tag{69}$$

Similarly, the system output is also the same as the desired output for the remaining sampling moments, i.e., $y_{k+1}(t) = y_d(t)$, for all $t \in \{1, \ldots, N\}$. As a result, the system will achieve perfect tracking of the desired trajectory in the first trial if $R = 0$.

When $\|R\| \to \infty$, we get $u_{k+1}(t) \to u_k(t)$ from Eq. (32), meaning the input does not change at all, so the convergence speed of the algorithm along the trial axis is reduced to zero. After the above analysis, it is clear that a smaller R leads to a faster convergence speed of the collaborative tracking error.

Indeed, the second term $\|\Delta u_{k+1}(t)\|_{\mathcal{R}}^2$ is also viewed as a penalty term that improves the robustness of the system to stochastic disturbances. A similar discussion can be found in Literature 1,11. The selection of the matrix R is considered as a trade-off between convergence speed and robustness.

Based on the previous analysis, the tuning guidelines regarding the weighting matrices Q, R and the penalty parameter ρ are given below.

1. Select Q: The selection of the matrix Q corresponds to the weights of the collaborative tracking error. It is usually set $Q = I$.

2. Select R: When stochastic disturbances are present, they can lead to fluctuations in the steady-state error, affecting the control accuracy. The selection of the matrix R needs to balance the convergence speed and the sensitivity to stochastic process disturbances. Start with $R = 0$, and then gradually increase r until the fluctuation of the steady-state error reaches the practical requirements or no longer decreases.

3. Select ρ: After ensuring the stability of the system, parameter ρ is tuned according to Eq. (64) to speed up the convergence of the control inputs of each subsystem. In addition, to reduce the computational effort, we can gradually reduce maximum ADMM iteration number $\vartheta_{\max}$ until the tracking error convergence performance appears to worsen.

The above guidelines provide a generalized tuning method for the proposed Algorithm 1, revealing the different effects of different parameters on the convergence performance. We will verify each of them in the simulations in the next section.

4.4 Computational Complexity Analysis

From the analysis in Section 2.2, the computational complexity of the lifted NOILC algorithm is $\mathcal{O}(N^3)$, which is clearly impractical for the collaborative tracking problem. The decentralized N-NOILC algorithm proposed in this paper does not use lifting techniques for system description, so it essentially avoids excessive computation. The computational complexity analysis of Algorithm 1 is given below.

In the time domain, during time index $t \to t+1$, Algorithm 1 is iterated $\vartheta_{\max}$ times according to the laws (25)-(27). The computational complexity of each iteration of the laws (25)-(27) is only related to the number of subsystems and the matrix dimension of subsystems. Thus the computational complexity of Algorithm 1 updated once can be written as $\mathcal{O}(s\vartheta_{\max}n_x^3)$, where n_x represents the subsystem matrix dimension. During one trial, the algorithm needs to be updated N times along the time axis, so the computational complexity is $\mathcal{O}(\vartheta_{\max}n_x^3N)$. For each subsystem, by choosing the optimal penalty parameter ρ it is possible to make the

$\vartheta_{\max}$ take on a small value, so that there is $\vartheta_{\max} \ll N$ in the long trial process. For the tracking task with long trial lengths, the computational complexity of each subsystem is only $\mathcal{O}(N)$.

Compared with the existing lifted NOILC algorithms, the computational complexity of the proposed N-NOILC algorithm is significantly reduced, and the required memory space is only proportional to the trial length N. A summary of the comparative results of the two approaches is given in Table 1.

Table 1: Comparison of computational cost.

Comparison	**N-NOILC**	**Lifted NOILC**
Computational complexity of a subsystem	$\mathcal{O}(\vartheta_{\max} n_x^3 N)$	$\mathcal{O}(\vartheta_{\max} n_x^3 N^3)$
Memory requirement for a subsystem	$\mathcal{O}(n_x^2 N)$	$\mathcal{O}(n_x^2 N^2)$
System global computational complexity	$\mathcal{O}(s\vartheta_{\max} n_x^3 N)$	$\mathcal{O}(s\vartheta_{\max} n_x^3 N^3)$
System global memory requirement	$\mathcal{O}(s n_x^2 N)$	$\mathcal{O}(s n_x^2 N^2)$

Remark 8. *In this paper, an increase in the number of subsystems helps reduce the output load of individual subsystems and enhance the overall scalability of the system. However, a larger number of subsystems means that the scale of the collaborative tracking problem becomes more complex, thereby slowing down ADMM convergence. In shared ADMM, a larger number of subsystems typically increases coordination complexity, requiring the algorithm to undergo more iterations $\vartheta_{\max}$ to achieve the same convergence accuracy. This is because the increase in subsystems amplifies the coupling effect between variables, raising the complexity of the coordination step and thus decreasing the convergence rate of the dual residual and the primal residual. As well, an increase in the number of subsystems leads to higher communication overhead in the system. In each iteration, all subsystems must exchange information about global shared variables and dual variables with the central node. If the number of subsystems grows, it is highly likely to exacerbate communication latency. Therefore, increasing the number of subsystems may not necessarily lead to better performance of the proposed algorithm.*

5 Simulation Validation

In order to validate the effectiveness of the proposed Algorithm 1, a heterogeneous networked dynamics system model consisting of seven different subsystems in [4] is adopted. The transfer function of the i^{th} subsystem is defined as

$$G_i(s) = \frac{5s-1}{s^2+5s+i},\ i = 1, \ldots, 7. \tag{70}$$

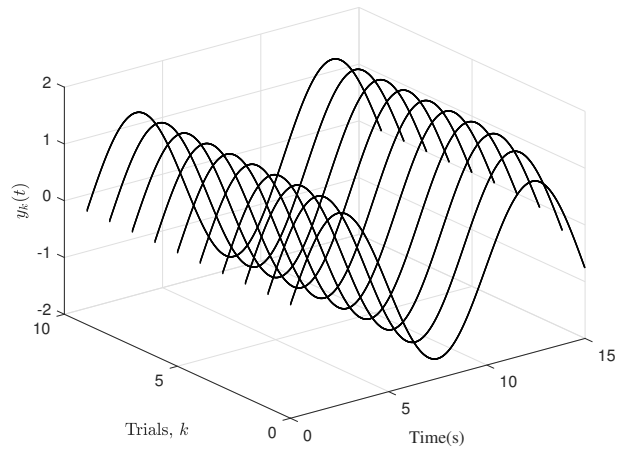

Figure 1: Outputs for the global system (3) on the different trial.

To obtain the discretized system model (1), the trial duration is set to 15s with a sampling time $T_s = 0.01s$, and hence the trial length $N = 1500$. The discretized state space matrices of the subsystem plant are as follows:

$$A_i = \begin{bmatrix} 0.951 & i \times 10^{-2} \\ 0.0098 & 1 \end{bmatrix},\ B_i = \begin{bmatrix} 0.0098 \\ 4.92 \times 10^{-5} \end{bmatrix},\ C_i = \begin{bmatrix} 5 & -1 \end{bmatrix},\ i = 1, \ldots, 7. \tag{71}$$

The desired trajectory for collaborative tracking is defined as

$$y_d(t) = 1.5 \sin 0.2\pi t,\ t \in [0, 15s]\,. \tag{72}$$

The input of the first trial is set to zero. In the following, the properties of the proposed decentralized N-NOILC algorithm are first validated.

5.1 Properties of the Decentralized N-NOILC Algorithm

In order to test the effectiveness of the proposed Algorithm 1, the output trajectory of the system is given, as shown in Fig. 1 to Fig. 3. We set the parameters $\rho = 1$, $Q = I$, $R = 0.01I$ and $\vartheta_{\max} = 8$. Fig. 1 illustrates the output trajectories of the global system on the different trial. Fig. 2 illustrates the output trajectories of the subsystem 1 on the different trial. Fig. 3 illustrates the output trajectories of the global system and some of the subsystems on the last trial. Due to the different dynamics of each subsystem, their output trajectories are also different. However, through the collaboration of all subsystems, the global output trajectory still achieves the perfect tracking of the desired trajectory.

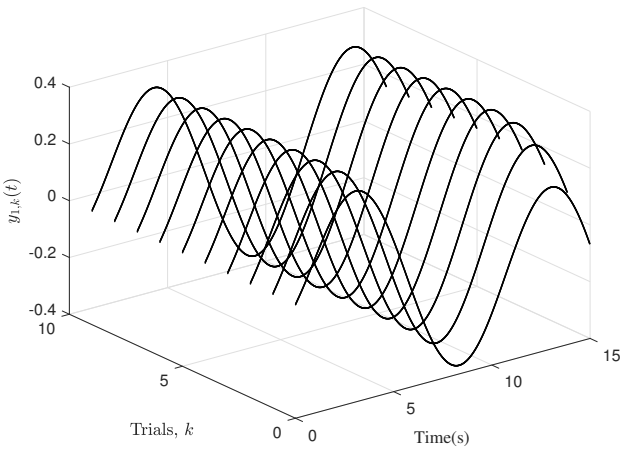

Figure 2: Outputs for the subsystem 1 on the different trial.

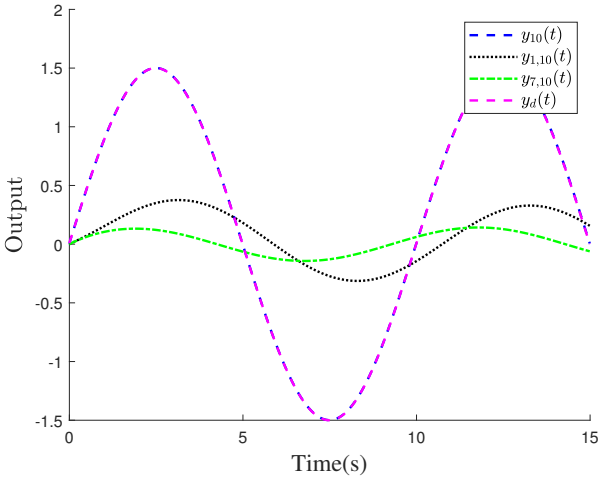

Figure 3: Outputs for the global system (3) and subsystems 1, 7 on the last trial.

The convergence behavior of Algorithm 1 is affected by several parameters, e.g., the maximum ADMM iteration number $\vartheta_{\max}$, the penalty parameter ρ, the weighting matrix R. The effects of these parameters are studied separately below.

To study the effect of $\vartheta_{\max}$, the value of all other parameters is fixed, with $\rho = 1$, $Q = I$, $R = 0.01I$. Fig. 4 shows that the performance criterion gradually converges to the optimal value with increasing $\vartheta_{\max}$, as described in Proposition 1. Fig. 5 illustrates how the collaborative tracking error changes as $\vartheta_{\max}$ increases.

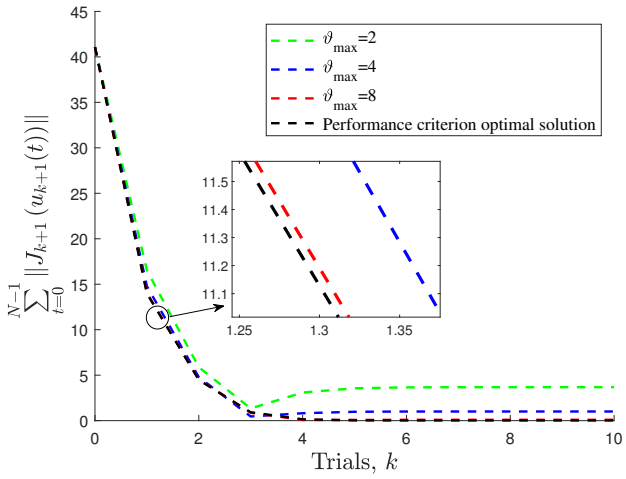

Figure 4: Performance criterion convergence results with different $\vartheta_{\max}$.

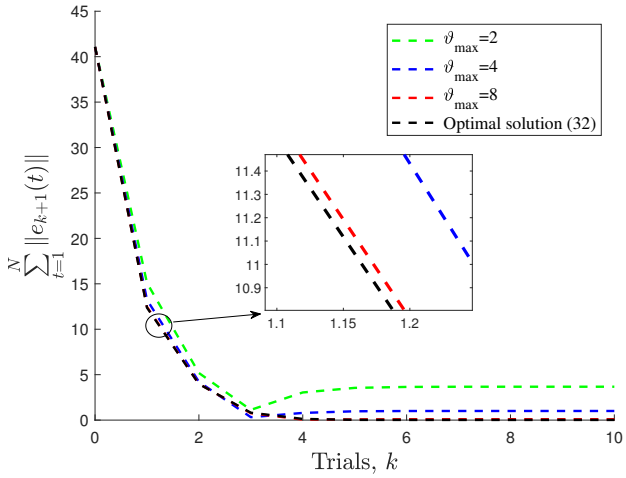

Figure 5: Error convergence results with different $\vartheta_{\max}$.

The collaborative tracking error norm is monotonically convergent along the trial axis, verifying Theorem 4.4. Moreover, as $\vartheta_{\max}$ increases, the variation of the error curve converges more and more to the optimal solution, verifying the equivalence in Eq. (28). After $\vartheta_{\max} \geq 8$, the decentralized convergence result is almost identical to the optimal solution, which means that a small iteration value is sufficient to achieve the desired convergence result, reflecting the efficient convergence ability of ADMM.

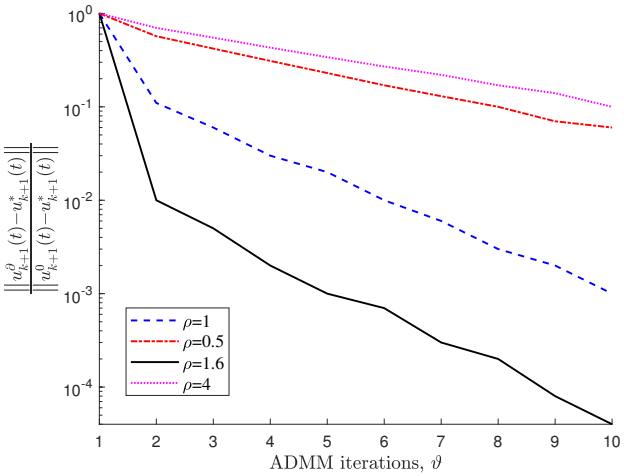

Figure 6: Input accuracy convergence speed with different ρ.

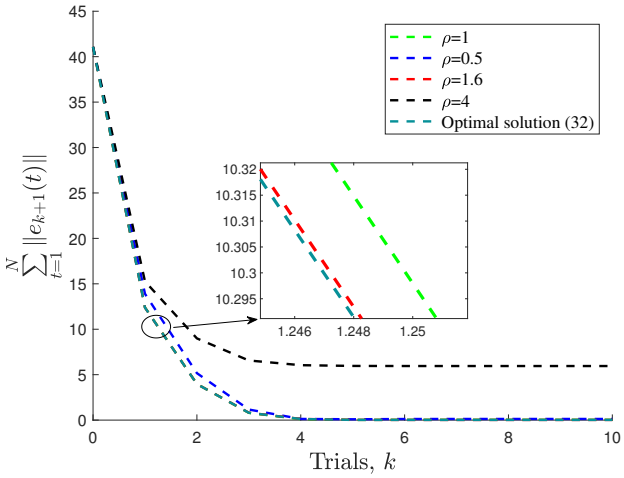

Figure 7: Error convergence results with different ρ.

Fig. 6 illustrates input convergence speed with different ρ in the first trial. We define the ADMM iteration convergence factor in (62) to characterize the input convergence speed, so as to find out the effect of changes in ρ on the convergence speed, where $u^{*}_{k+1}(t)$ is the optimal input obtained by (32). A smaller convergence factor means that the ADMM iterations converge faster. Based on the conclusion of Proposition 2, we calculate the optimal penalty parameter $\rho^{*} = 1.6$. Increasing ρ gradually from 0.5 to 4, the input convergence speed is first faster and then slower.

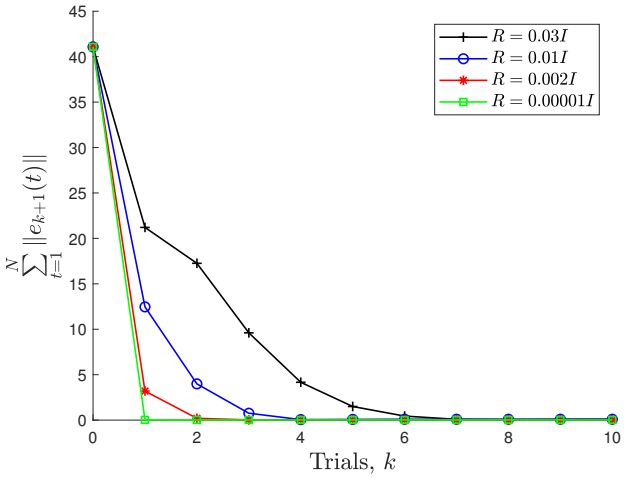

Figure 8: Error convergence speed with different R ($Q = I$).

Clearly, convergence result with $\rho = 1.6$ is the most ideal. Also, we investigate the effect of choosing ρ on the collaborative tracking error. Fig. 7 provides the error convergence results with different ρ, where setting $\vartheta_{\max} = 10$. The convergence result with $\rho = 1.6$ is the closest to the global optimum. For the case $\rho = 4$, a further increase to $\vartheta_{\max}$ is required to improve the convergence performance.

Fig. 8 illustrates the effect of different R on the error convergence properties. The other parameters are fixed, with $\vartheta_{\max} = 8$, $\rho = 1.6$, $Q = I$. The change of R has no effect on the convergence accuracy of the proposed algorithm because the steady-state error of each curve converges to zero along the trial axis. In addition, as R decreases, the error converges faster. These validate the conclusions of Section 4.2 and 4.3.

To study the effect of the process disturbances described in Remark 7 on the convergence of the algorithm, we add stochastic disturbances $d_{i,k}(t) = 0.001*$ **randn** to the state of each subsystem, where **randn** is a stochastic disturbance that satisfies the standard normal distribution. Fig. 9 shows the system output result with stochastic disturbance on the different trial. Fig. 10 shows the error convergence comparison between disturbance and nominal system. In the presence of disturbances, the system error cannot converge to 0 but instead fluctuates within an interval related to the magnitude of the disturbances.

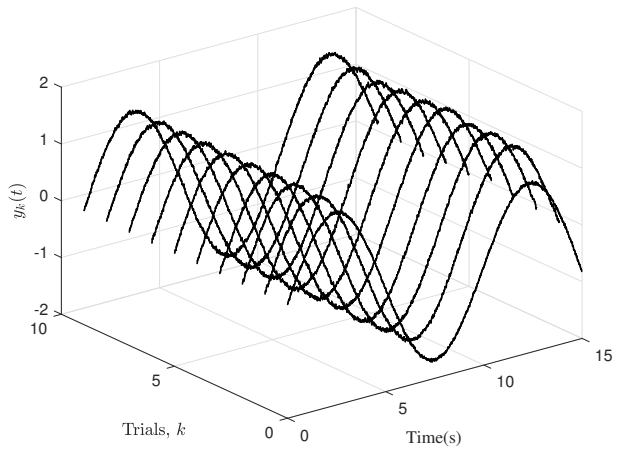

Figure 9: Output result with stochastic disturbance on the different trial.

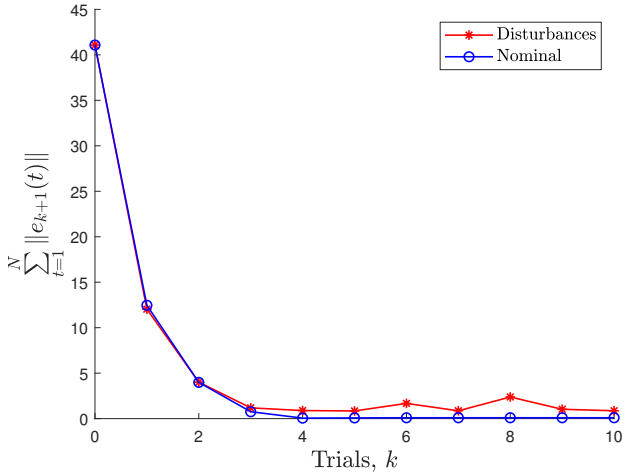

Figure 10: Error convergence comparison between disturbance and nominal system.

5.2 Scalability of the Decentralized N-NOILC Algorithm

For decentralized algorithms, scalability refers to the ability of the algorithm to effectively adapt and handle changes in the whole system size (i.e., the number of subsystems) when such changes occur. To test scalability, we consider the case of adding and removing a subsystem between trials 3 and 4, respectively. The

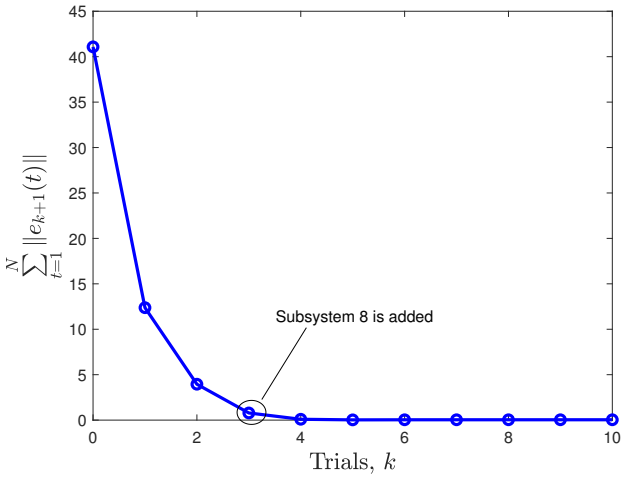

Figure 11: Error convergence result for adding subsystem on the 3rd trial.

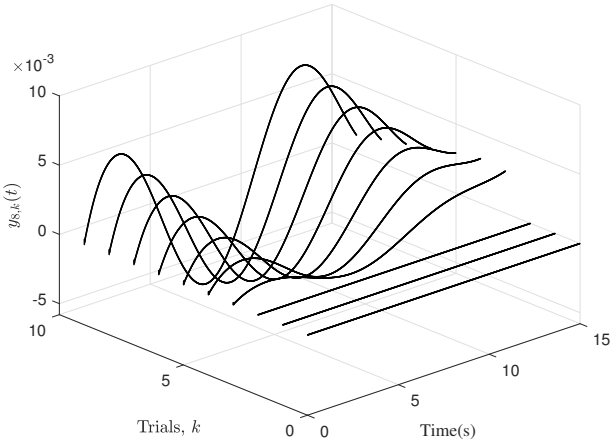

Figure 12: Outputs for the added subsystem 8 on the different trial.

parameters are fixed when the network size changes, with $\vartheta_{\max} = 8$, $\rho = 1.6$, $Q = I$, $R = 0.01I$.

The error convergence result for adding subsystem is shown in Fig. 11. The transfer function for the newly added subsystem 8 is

$$G_8(s) = \frac{5s - 1}{s^2 + 5s + 8}.$$

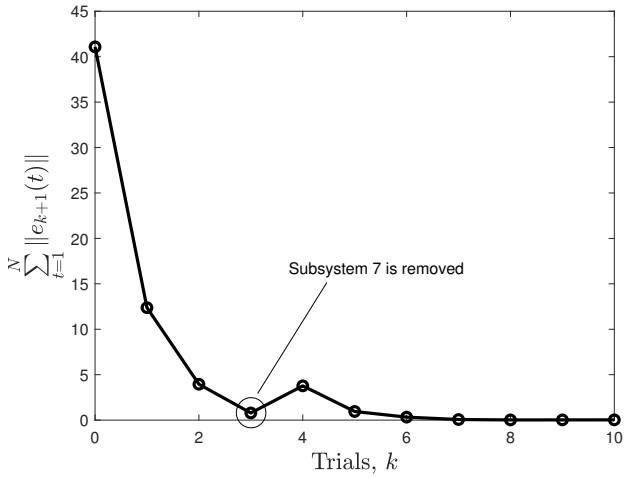

Figure 13: Error convergence result for removing subsystem on the 3rd trial.

After subsystem 8 is added, the error profile remains monotonically convergent. This means that the proposed algorithm enables subsystem 8 to quickly integrate into the whole system to drive collaborative tracking with other subsystems. Increasing the number of subsystems helps the whole system to accomplish the collaborative tracking task. Fig. 12 provides the output trajectories of the subsystem 8 on each trial.

The error convergence result for removing subsystem is shown in Fig. 13. During the first three trials, the error converges quickly as usual. On the third trial, subsystem 7 is removed from the whole system, causing a transient error. However, the proposed algorithm immediately adjusts the input of the remaining 6 subsystems to cope with the network change, so that the collaborative tracking error quickly monotonically converges again to zero after the next few trials.

For the proposed decentralized N-NOILC algorithm, by simply activating/deactivating the local inputs of subsystem 8/7, the algorithm enables the system to continue to update automatically until the error converges (other parameters are not retuned), fully reflecting the excellent scalability. The proposed algorithm does not need to retune any parameter to ensure error convergence when the whole system size changes.

In contrast, some existing ILC designs for collaborative tracking have poor scalability, for example, the algorithm with centralized control architecture in 4 need to redesign control laws for central controller, and the decentralized algorithms in 5 may require recalculation of learning gains γ that satisfy convergence conditions.

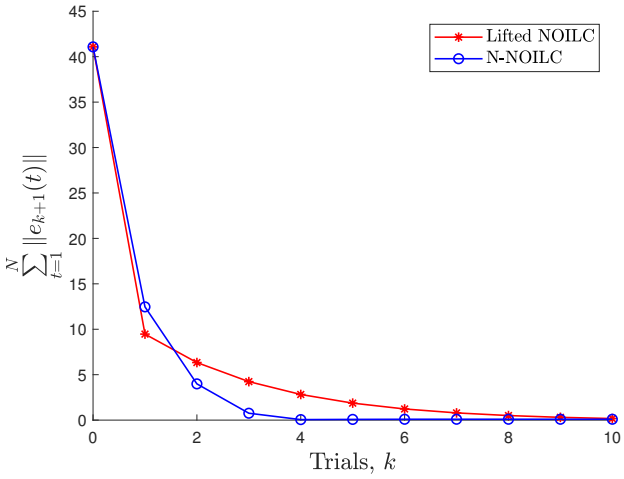

Figure 14: Error convergence comparison between N-NOILC and lifted NOILC.

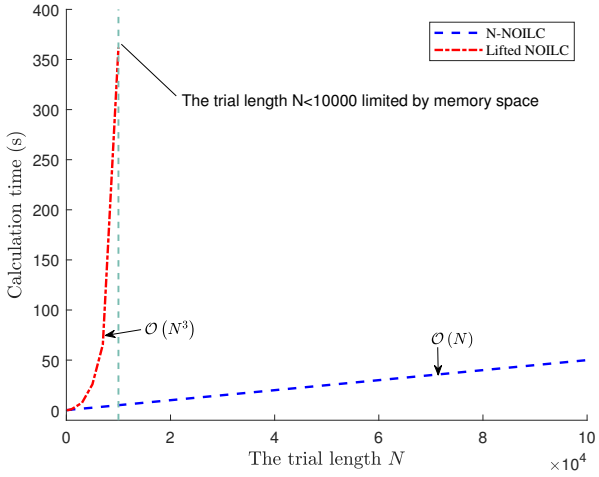

Figure 15: Calculation time comparison between N-NOILC and lifted NOILC.

5.3 Comparison of N-NOILC With Existing Methods

Case 1:

The N-NOILC algorithm proposed in this paper can be regarded as an improved algorithm with efficient computation of the lifted NOILC algorithm in 4, so a comparison of the two is given below. Fig. 14 illustrates the error convergence comparison between decentralized N-NOILC and lifted NOILC. Setting same pa-

rameters, both algorithms achieve good control of the model (70). Thanks to the fact that they both introduce vector 2-norm and model information to find the control gain, they both achieve monotonic convergence of the collaborative tracking error.

Fig. 15 compares the calculation time of the two algorithms. This is a key metric that characterizes the computational cost. The runtime environment for the comparison simulation is MATLAB R2021a on a laptop with 16GB RAM and 12^{th} Core i5 processor. Both algorithms set the same number of ADMM iterations, with $\vartheta_{\max} = 8$. The calculation time spent by them during one trial is counted separately by setting different trial lengths.

The difference between the two algorithms is very obvious. For the lifted NOILC algorithm, its high computational cost arises mainly from the operation of high-dimensional lifted matrices and supervectors. The red curve shows that its calculation time grows dramatically as the trial length N increases. In addition, the trial length needs to satisfy $N < 10000$, otherwise our computer will not be able to provide enough Random Access Memory (RAM) for the algorithm implementation. If the sampling frequency is set to 1KHz, then the duration time of each trial must be limited to 10s. Also, the memory space required to store the lifted matrix is proportional to the square of the trial length, i.e., $N \times N$. All of the above place a limit on the sampling frequency and trial duration.

In contrast, the computation time of proposed N-NOILC algorithm grows linearly with trial length, which relaxes the constraints on trial duration and sampling frequency. This is because the proposed approach does not enable the lifting technique, which avoids the construction and computation of high-dimensional matrices and reduces the computational complexity from the root. The above comparative results show that the proposed N-NOILC effectively relaxes the limitation of computer RAM on the lifted NOILC, greatly shortens the computation time, and significantly improves the computational efficiency.

Case 2:

Except for the proposed N-NOILC, frequency domain NOILC also helps reduce computational complexity, although the research on such schemes has focused on improving robust monotonic convergence. In the frequency domain, system dynamics are described by the discrete-time transfer function $G(z)$, and the update law is expressed as complex multiplications at each frequency point as follows [9]:

$$u_{k+1}(z) = Q(z)u_k(z) + L(z)e_k(z). \tag{73}$$

Notably, since the z-transform is defined over an infinite time horizon, i.e., $N = \infty$, the z-domain representation is an approximation of the NOILC system. The main steps of frequency domain NOILC are time-frequency transformation ($\mathcal{O}(N\log_2 N)$), frequency domain filtering ($\mathcal{O}(N)$), and frequency-time transformation ($\mathcal{O}(N\log_2 N)$),

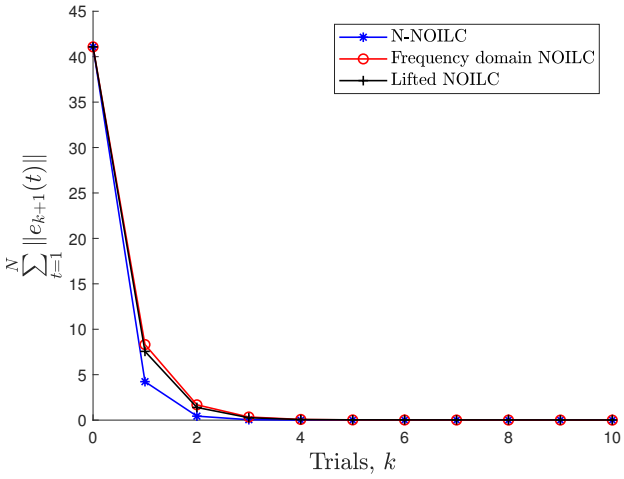

Figure 16: Error convergence comparison between N-NOILC, frequency domain NOILC and lifted NOILC.

thus its overall computational complexity is expressed as $\mathcal{O}(N\log_2 N)$, which is significantly lower than $\mathcal{O}(N^3)$ of time domain lifted NOILC.

Next, we select the numerical model from Reference 9 for simulation and compare it with the proposed N-NOILC. Since there is currently no relevant research on frequency domain NOILC for collaborative tracking scenarios, we only conduct simulations on the tracking problem of a single system here to study the differences between different NOILC methods. Consider a system with the following continuous-time transfer function [9]:

$$G(s) = \frac{s+20}{s+5}.$$

Both the system sampling time and the desired trajectory are as described above. Fig. 16 illustrates the error convergence comparison between N-NOILC, frequency domain NOILC and lifted NOILC. For frequency domain NOILC in 9, the weighting parameters are set to $\lambda = 2$, $\beta = 0$. Thanks to norm optimization framework, the three methods achieve monotonic convergence of the tracking error.

Fig. 17 counts the calculation time of them during one trial. As the trial length N increases, N-NOILC and frequency domain NOILC require significantly less calculation time than lifted NOILC. Interestingly, although the computational complexity of N-NOILC ($\mathcal{O}(N)$) is lower than that of frequency domain NOILC ($\mathcal{O}(N\log_2 N)$), N-NOILC requires more computational time than frequency domain NOILC when the trial length exceeds a certain value. This is because when performing Fourier transforms, MATLAB's built-in **fft** function is highly opti-

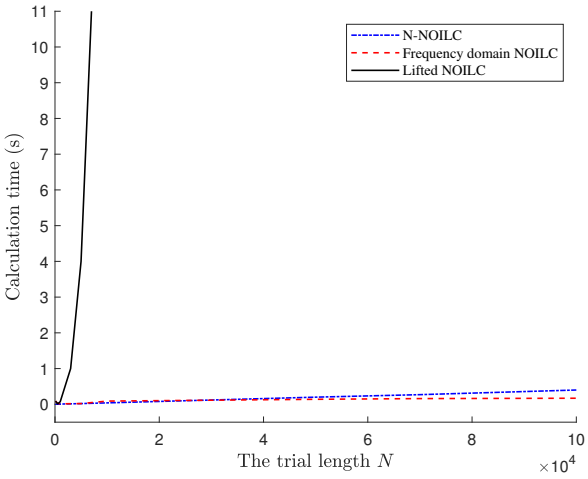

Figure 17: Calculation time comparison between N-NOILC, frequency domain NOILC and lifted NOILC.

mized based on the computer's underlying optimized libraries, enabling real-time processing of million-point signals with extremely high computational efficiency. In contrast, the time domain N-NOILC relies on loop operations, thus the calculation time is longer than that of frequency domain operations in MATLAB. Therefore, how to further accelerate the calculation time of N-NOILC becomes an open problem in the future.

6 Conclusions and Future Work

In this paper, a new decentralized N-NOILC approach for collaborative tracking is presented. A clear benefit is that the computational cost of lifted NOILC is significantly reduced by avoiding enabling lifting technique, including less computation time and memory consumption. This provides more freedom in the selection of trial length N. For the specific collaborative tracking problem, we introduce the decentralized ADMM, which enables the algorithm to be deployed in a decentralized manner to every subsystem. This parallel computation scheme improves the execution efficiency and scalability of the algorithm while without sacrificing the global optimal control performance of the whole system at the same time. The proposed approach is particularly suitable for large-scale systems and long-time tracking tasks with high sampling frequencies.

The proposed decentralized N-NOILC approach is essentially a model-based scheme meaning that the performance of the algorithm is affected by the model-

ing accuracy. Future work will be devoted to robustness analysis and design for model uncertainty. Even further, a data-driven N-NOILC approach is extremely attractive. In addition, the main contribution of the paper is to reduce the computational complexity of lifted NOILC. However, the test in Theorem 4.5 to check whether the proposed algorithm satisfies the monotonic convergence condition still requires the construction of a few high-dimensional lifted matrices. Therefore, exploring and investigating the monotonic convergence condition independent of the trial length N is a worthy consideration.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62361136585), the 111 Project (B23008), the National Science Centre in Poland (2023/48/Q/ST7/00205), the Serbian Ministry of Science, Technological Development and Innovation (No. 451-03-47/2023-01/200108).

References

[1] Kira L Barton and Andrew G Alleyne. A norm optimal approach to time-varying ilc with application to a multi-axis robotic testbed. *IEEE Transactions on Control Systems Technology*, 19(1):166–180, 2010.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

[3] Xuhui Bu, Jiaqi Liang, Zhongsheng Hou, and Ronghu Chi. Data-driven terminal iterative learning consensus for nonlinear multiagent systems with output saturation. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1963–1973, 2021.

[4] Bin Chen and Bing Chu. Decentralized iterative learning control for constrained collaborative tracking. *International Journal of Robust and Nonlinear Control*, 33(7):3988–4008, 2023.

[5] Shangcheng Chen and Christopher T Freeman. A decentralised iterative learning control framework for collaborative tracking. *Mechatronics*, 72:102465, 2020.

[6] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66:889–916, 2016.

[7] Santosh Devasia. Iterative learning control with time-partitioned update for collaborative output tracking. *Automatica*, 69:258–264, 2016.

[8] Luyuan Gao, Zhihe Zhuang, Hongfeng Tao, Yiyang Chen, and Vladimir Stojanovic. Non-lifted norm optimal iterative learning control for networked dynamical systems: A computationally efficient approach. *Journal of the Franklin Institute*, 361(15):107112, 2024.

[9] Xinyi Ge, Jeffrey L Stein, and Tulga Ersal. Frequency-domain analysis of robust monotonic convergence of norm-optimal iterative learning control. *IEEE Transactions on Control Systems Technology*, 26(2):637–651, 2017.

[10] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

[11] Aleksandar Haber, Rufus Fraanje, and Michel Verhaegen. Linear computational complexity robust ILC for lifted systems. *Automatica*, 48(6):1102–1110, 2012.

[12] WBJ Hakvoort, Ronald GKM Aarts, Johannes van Dijk, and Jan B Jonker. Lifted system iterative learning control applied to an industrial robot. *Control Engineering Practice*, 16(4):377–391, 2008.

[13] WBJ Hakvoort, Ronald GKM Aarts, Johannes van Dijk, and Jan B Jonker. A computationally efficient algorithm of iterative learning control for discrete-time linear time-varying systems. *Automatica*, 45(12):2925–2929, 2009.

[14] Shumaila Javaid, Nasir Saeed, Zakria Qadir, Hamza Fahim, Bin He, Houbing Song, and Muhammad Bilal. Communication and control in collaborative UAVs: recent advances and future trends. *IEEE Transactions on Intelligent Transportation Systems*, 24(6):5719–5739, 2023.

[15] Haruhisa Kawasaki, Satoshi Ueki, and Satoshi Ito. Decentralized adaptive coordinated control of multiple robot arms without using a force sensor. *Automatica*, 42(3):481–488, 2006.

[16] Ying Kong, Xiao-Dong Li, and Xuefang Li. Iterative learning control approach for linear discrete time-varying systems with different initial time points. *Journal of the Franklin Institute*, 361(6):106702, 2024.

[17] Richard W Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73(10):930–954, 2000.

[18] Jingyi Lu, Zhixing Cao, Qinran Hu, Zuhua Xu, Wenli Du, and Furong Gao. Optimal iterative learning control for batch processes in the presence of time-varying dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(1):680–692, 2022.

[19] Mikael Norrlöf and Svante Gunnarsson. Time and frequency domain convergence properties in iterative learning control. *International Journal of Control*, 75(14):1114–1126, 2002.

[20] Daniel P Palomar and Yonina C Eldar. *Convex optimization in signal processing and communications*. Cambridge University Press, 2010.

[21] Jonathan Realmuto, Rahul B Warrier, and Santosh Devasia. Data-inferred personalized human-robot models for iterative collaborative output tracking. *Journal of Intelligent & Robotic Systems*, 91:137–153, 2018.

[22] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

[23] Doganay Sirintuna, Alberto Giammarino, and Arash Ajoudani. An object deformation-agnostic framework for human–robot collaborative transportation. *IEEE Transactions on Automation Science and Engineering*, pages 1–14, 2023.

[24] Tong Duy Son, Goele Pipeleers, and Jan Swevers. Robust monotonic convergent iterative learning control. *IEEE Transactions on Automatic Control*, 61(4):1063–1068, 2015.

[25] Fazhi Song, Yang Liu, Dong Shen, Li Li, and Jiubin Tan. Learning control for motion coordination in wafer scanners: Toward gain adaptation. *IEEE Transactions on Industrial Electronics*, 69(12):13428–13438, 2022.

[26] Heqing Sun and Andrew G Alleyne. A computationally efficient norm optimal iterative learning control approach for LTV systems. *Automatica*, 50(1):141–148, 2014.

[27] Andre Teixeira, Euhanna Ghadimi, Iman Shames, Henrik Sandberg, and Mikael Johansson. The admm algorithm for distributed quadratic problems: Parameter selection and constraint preconditioning. *IEEE Transactions on Signal Processing*, 64(2):290–305, 2015.

[28] DABM Tharayil and Andrew G Alleyne. A survey of iterative learning control: A learning-based method for high performance tracking control. *IEEE Control Systems Magazine*, 26(3):96–114, 2006.

[29] Jeroen Van de Wijdeven, Tijs Donkers, and Okko Bosgra. Iterative learning control for uncertain systems: Robust monotonic convergence analysis. *Automatica*, 45(10):2383–2391, 2009.

[30] Jurgen van Zundert, Joost Bolder, Sjirk Koekebakker, and Tom Oomen. Resource-efficient ILC for LTI/LTV systems through LQ tracking and stable inversion: Enabling large feedforward tasks on a position-dependent printer. *Mechatronics*, 38:76–90, 2016.

[31] Zhijun Wang, Tengfei Hu, and Lijun Long. Multi-UAV safe collaborative transportation based on adaptive control barrier function. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53:6975–6983, 2023.

[32] Chenhui Zhou, Hongfeng Tao, Yiyang Chen, Vladimir Stojanovic, and Wojciech Paszke. Robust point-to-point iterative learning control for constrained systems: A minimum energy approach. *International Journal of Robust and Nonlinear Control*, 32(18):10139–10161, 2022.

[33] Zhihe Zhuang, Hongfeng Tao, Yiyang Chen, Vladimir Stojanovic, and Wojciech Paszke. An optimal iterative learning control approach for linear systems with nonuniform trial lengths under input constraints. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(6):3461–3473, 2023.