1

Introduction

1.1 Introduction to Iterative Learning Control

Iterative learning control (ILC), as an effective control strategy, is designed to improve current control performance for unpredictable systems by fully utilizing past control experience. Specifically, ILC is designed for systems that complete tasks over a fixed time interval and perform them repeatedly. The underlying philosophy mimics the human learning process that "practice makes perfect." By synthesizing control inputs from previous control inputs and tracking errors, the controller is able to learn from past experience and improve current tracking performance. ILC was initially developed by Arimoto *et al.* (1984), and has been widely explored by the control community since then (Moore, 1993; Bien and Xu, 1998; Chen and Wen, 1999; Longman, 2000; Norrlof and Gunnarsson, 2002; Xu and Tan, 2003; Bristow *et al.*, 2006; Moore *et al.*, 2006; Ahn *et al.*, 2007a; Rogers *et al.*, 2007; Ahn *et al.*, 2007b; Xu *et al.*, 2008; Wang *et al.*, 2009, 2014).

Figure 1.1 shows the schematic diagram of an ILC system, where the subscript i denotes the iteration index and y_d denotes the reference trajectory. Based on the input signal, u_i, at the ith iteration, as well as the tracking error $e_i = y_d - y_i$, the input u_{i+1} for the next iteration, namely the $(i+1)$th iteration, is constructed. Meanwhile, the input signal u_{i+1} will also be stored into memory for use in the $(i+2)$th iteration. It is important to note that in Figure 1.1, a closed feedback loop is formed in the iteration domain rather than the time domain. Compared to other control methods such as proportional-integral-derivative (PID) control and sliding mode control, there are a number of distinctive features about ILC. First, ILC is designed to handle repetitive control tasks, while other control techniques don't typically take advantage of task repetition—under a repeatable control environment, repeating the same feedback would yield the same control performance. In contrast, by incorporating learning, ILC is able to improve the control performance iteratively. Second, the control objective is different. ILC aims at achieving perfect tracking over the whole operational interval. Most control methods aim to achieve asymptotic convergence in tracking accuracy over time. Third, ILC is a feedforward control method if viewed in the time domain. The plant shown in Figure 1.1 is a generalized plant, that is, it can actually include a feedback loop. ILC can be used to further improve the performance of the generalized plant. As such, the generalized plant could be made stable in the time domain, which is helpful in guaranteeing transient response while learning takes place. Last but not

Iterative Learning Control for Multi-agent Systems Coordination, First Edition.
Shiping Yang, Jian-Xin Xu, Xuefang Li, and Dong Shen.

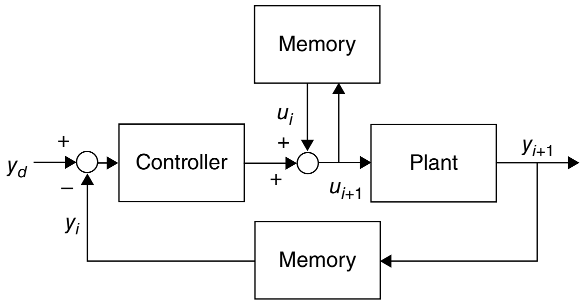

Figure 1.1 The framework of ILC.

least, ILC is a partially model-free control method. As long as an appropriate learning gain is chosen, perfect tracking can be achieved without using a perfect plant model.

Generally speaking, there are two main frameworks for ILC, namely contraction-mapping (CM)-based and composite energy function (CEF)-based approaches. A CM-based iterative learning controller has a very simple structure and is easy to implement. A correction term in the controller is constructed from the output tracking error; to ensure convergence, an appropriate learning gain is selected based on system gradient information in place of an accurate dynamic model. As a partially model-free control method, CM-based ILC is applicable to non-affine-in-input systems. These features are highly desirable in practice as there are plenty of data available in industry processes but there is a shortage of accurate system models. CM-based ILC has been adopted in many applications, for example X-Y tables, chemical batch reactors, laser cutting systems, motor control, water heating systems, freeway traffic control, wafer manufacturing, and so on (Ahn *et al.*, 2007a). A limitation of CM-based ILC is that it is only applicable to global Lipschitz continuous (GLC) systems. The GLC condition is required by ILC in order to form a contractive mapping, and rule out the finite escape time phenomenon. In comparison, CEF-based ILC, a complementary approach to CM-based ILC, applies a Lyapunov-like method to design learning rules. CEF is an effective method to handle locally Lipschitz continuous (LLC) systems, because system dynamics is used in the design of learning and feedback mechanisms. It is, however, worthwhile pointing out that in CM-based ILC, the learning mechanism only requires output signals, while in CEF-based ILC, full state information is usually required. CEF-based ILC has been applied in satellite trajectory keeping (Ahn *et al.*, 2010) and robotic manipulator control (Tayebi, 2004; Tayebi and Islam, 2006; Sun *et al.*, 2006).

This book follows the two main frameworks and investigates the multi-agent coordination problem using ILC. To illustrate the underlying idea and properties of ILC, we start with a simple ILC system.

Consider the following linear time-invariant dynamics:

$$\dot{x}_i(t) = ax_i(t) + u_i(t), \ t \in [0, T], \tag{1.1}$$

where i is the iteration index, a is an unknown constant parameter, and T is the trial length. Let the target trajectory be $x_d(t)$, which is generated by

$$\dot{x}_d(t) = ax_d(t) + u_d(t), \ t \in [0, T], \tag{1.2}$$

with $u_d(t)$ is the desired control signal. The control objective is to tune $u_i(t)$ such that without any prior knowledge about the parameter a, the tracking error

$e_i(t) \triangleq x_d(t) - x_i(t)$ can converge to zero as the iteration number increases, that is, $\lim_{i\to\infty} e_i(t) = 0$ for $t \in [0, T]$.

We perform the ILC controller design and convergence analysis for this simple control problem under the frameworks of both CM-based and CEF-based approaches, in order to illustrate the basic concepts in ILC and analysis techniques. To restrict our discussion, the following assumptions are imposed on the dynamical system (1.1).

Assumption 1.1 The identical initialization condition holds for all iterations, that is, $x_i(0) = x_d(0)$, $\forall i \in \mathbb{N}$.

Assumption 1.2 For $\forall x_d(t)$, $t \in [0, T]$, there exists a $u_d(t)$, $t \in [0, T]$ such that $u_i(t) \to u_d(t)$ implies $x_i(t) \to x_d(t)$, $t \in [0, T]$.

1.1.1 Contraction-Mapping Approach

Under the framework of CM-based methodology, we apply the following D-type updating law to solve the trajectory tracking problem:

$$u_{i+1} = u_i + \gamma \dot{e}_i, \tag{1.3}$$

where $\gamma > 0$ is the learning gain to be determined. Our objective is to show that the ILC law (1.3) can converge to the desired u_d, which implies the convergence of the tracking error $e_i(t)$, $t \in [0, T]$ as i increases.

Define $\Delta u_i = u_d - u_i$. First we can derive the relation

$$\begin{aligned}\Delta u_{i+1} &= u_d - u_{i+1} \\ &= u_d - u_i - \gamma \dot{e}_i \\ &= \Delta u_i - \gamma \dot{e}_i.\end{aligned} \tag{1.4}$$

Furthermore, the state error dynamics is given by

$$\begin{aligned}\dot{e}_i &= \dot{x}_d - \dot{x}_i \\ &= (a x_d + u_d) - (a x_i + u_i) \\ &= a e_i + \Delta u_i.\end{aligned} \tag{1.5}$$

Combining (1.4) and (1.5) gives:

$$\begin{aligned}\Delta u_{i+1} &= \Delta u_i - \gamma \dot{e}_i \\ &= (1-\gamma)\Delta u_i - a\gamma e_i.\end{aligned} \tag{1.6}$$

Integrating both sides of the state error dynamics and using Assumption 1.1 yields

$$\begin{aligned}e_i(t) &= e_i(0) + \int_0^t e^{a(t-\tau)} \Delta u_i(\tau) d\tau \\ &= \int_0^t e^{a(t-\tau)} \Delta u_i(\tau) d\tau.\end{aligned} \tag{1.7}$$

Then, substituting (1.7) into (1.6), we obtain

$$\Delta u_{i+1} = (1-\gamma)\Delta u_i - a\gamma \int_0^t e^{a(t-\tau)} \Delta u_i(\tau) d\tau. \tag{1.8}$$

Taking λ-norm on both sides of (1.8) gives

$$\begin{aligned}|\Delta u_{i+1}|_\lambda &\leq |1-\gamma||\Delta u_i|_\lambda + a\gamma\frac{1-e^{-(\lambda-a)T}}{\lambda-a}|\Delta u_i|_\lambda\\ &\triangleq \rho_1|\Delta u_i|_\lambda,\end{aligned} \tag{1.9}$$

where $\rho_1 \triangleq |1-\gamma| + a\gamma\frac{1-e^{-(\lambda-a)T}}{\lambda-a}$, and the λ-norm is defined as

$$|\Delta u_{i+1}|_\lambda = \sup_{t\in[0,T]} e^{-\lambda t}|\Delta u_{i+1}(t)|.$$

The λ-norm is just a time weighted norm and is used to simplify the derivation. It will be formally defined in Section 1.4.

If $|1-\gamma| < 1$ in (1.9), it is possible to choose a sufficiently large $\lambda > a$ such that $\rho_1 < 1$. Therefore, (1.9) implies that $\lim_{t\to\infty}|\Delta u_i|_\lambda = 0$, namely $\lim_{t\to\infty} u_i(t) = u_d(t),\ t\in[0,T]$.

1.1.2 Composite Energy Function Approach

In this subsection, the ILC controller will be developed and analyzed under the framework of CEF-based approach. First of all, the error dynamics of the system (1.1) can be expressed as follows:

$$\dot{e}_i(t) = -ax_i + \dot{x}_d - u_i, \tag{1.10}$$

where x_d is the target trajectory.

Let k be a positive constant. By applying the control law

$$u_i = -ke_i + \dot{x}_d - \hat{a}_i(t)x_i \tag{1.11}$$

and the parametric updating law $\forall t\in[0,T]$,

$$\hat{a}_i(t) = \hat{a}_{i-1}(t) + x_ie_i, \quad \hat{a}_{-1}(t) = 0, \tag{1.12}$$

we can obtain the convergence of the tracking error e_i as i tends to infinity.

In order to facilitate the convergence analysis of the proposed ILC scheme, we introduce the following CEF:

$$E_i(t) = \frac{1}{2}e_i^2(t) + \frac{1}{2}\int_0^t \phi_i^2(\tau)d\tau, \tag{1.13}$$

where $\phi_i(t) \triangleq \hat{a}_i - a$ is the estimation error of the unknown parameter a.

The difference of E_i is

$$\begin{aligned}\Delta E_i(t) &= E_i - E_{i-1}\\ &= \frac{1}{2}e_i^2 + \frac{1}{2}\int_0^t(\phi_i^2 - \phi_{i-1}^2)d\tau - \frac{1}{2}e_{i-1}^2.\end{aligned} \tag{1.14}$$

By using the identical initialization condition as in Assumption 1.1, the error dynamics (1.10), and the control law (1.11), the first term on the right hand side of (1.14) can be calculated as

$$\begin{aligned}\frac{1}{2}e_i^2 &= \int_0^t e_i\dot{e}_id\tau\\ &= \int_0^t e_i(-\dot{x}_d + ax_i + u_i)d\tau\\ &= \int_0^t(-\phi_ix_ie_i - ke_i^2)d\tau.\end{aligned} \tag{1.15}$$

In addition, the second term on the right hand side of (1.14) can be expressed as

$$\frac{1}{2}\int_0^t (\phi_i^2 - \phi_{i-1}^2)d\tau = \frac{1}{2}\int_0^t (\hat{a}_{i-1} - \hat{a}_i)(2a - 2\hat{a}_i + \hat{a}_i - \hat{a}_{i-1})d\tau$$
$$= \int_0^t (\phi_i x_i e_i - \frac{1}{2}x_i^2 e_i^2)d\tau, \tag{1.16}$$

where the updating law (1.12) is applied. Clearly, $\phi_i x_i e_i$ appears in (1.15) and (1.16) with opposite signs. Combining (1.14), (1.15), and (1.16) yields

$$\Delta E_i(t) = -k\int_0^t e_i^2 d\tau - \frac{1}{2}\int_0^t x_i^2 e_i^2 d\tau - \frac{1}{2}e_{i-1}^2$$
$$\leq -\frac{1}{2}e_{i-1}^2 < 0. \tag{1.17}$$

The function E_i is a monotonically decreasing sequence, hence is bounded if E_0 is bounded.

Now, let us show the boundedness of E_0. For the linear plant (1.1) or in general GLC plants, there will be no finite escape time, thus E_0 is bounded. For local Lipschitz continuous plants, ILC designed under CEF guarantees there is no finite escape time (see Xu and Tan, 2003, chap. 7), thus E_0 is bounded. Hence, the boundedness of $E_0(t)$ over $[0, T]$ is obtained.

Consider a finite sum of ΔE_i,

$$\sum_{j=1}^{i} \Delta E_j = \sum_{j=1}^{i}(E_j - E_{j-1}) = E_i - E_0, \tag{1.18}$$

and apply the inequality (1.17); we have:

$$E_i(t) = E_0(t) + \sum_{j=1}^{i} \Delta E_j$$
$$\leq E_0(t) - \frac{1}{2}\sum_{j=1}^{i} e_{j-1}^2. \tag{1.19}$$

Because of the positiveness of E_i and boundedness of E_0, $e_i(t)$ converges to zero in a pointwise fashion as i tends to infinity.

1.2 Introduction to MAS Coordination

In the past several decades, MAS coordination and control problems have attracted considerable attention from many researchers of various backgrounds due to their potential applications and cross-disciplinary nature. *Consensus* in particular is an important class of MAS coordination and control problems (Cao *et al.*, 2013). According to Olfati-Saber *et al.* (2007), in networks of agents (or dynamic systems), consensus means to reach an agreement regarding certain quantities of interest that are associated with all agents. Depending on the specific application, these quantities could be velocity, position, temperature, orientation, and so on. In a consensus realization, the control action of an agent is generated based on the information received or measured from its neighborhood.

Since the control law is a kind of distributed algorithm, it is more robust and scalable compared to centralized control algorithms.

The three main components in MAS coordination are the agent model, the information sharing topology, and the control algorithm or consensus algorithm.

Agent models range from simple single integrator model to complex nonlinear models. Consensus results on single integrators are reported by Jadbabaie *et al.* (2003), Olfati-Saber and Murray (2004), Moreau (2005), Ren *et al.* (2007), and Olfati-Saber *et al.* (2007). Double integrators are investigated in Xie and Wang (2005), Hong *et al.* (2006), Ren (2008a), and Zhang and Tian (2009). Results on linear agent models can be found in Xiang *et al.* (2009), Ma and Zhang (2010), Li *et al.* (2010), Huang (2011), and Wieland *et al.* (2011). Since the Lagrangian system can be used to model many practical systems, consensus has been extensively studied by means of the Lagrangian system. Some representative works are reported by Hou *et al.* (2009), Chen and Lewis (2011), Mei *et al.* (2011), and Zhang *et al.* (2014).

Information sharing among agents is one of the indispensable components for consensus seeking. Information sharing can be realized by direct measurement from on board sensors or communication through wireless networks. The information sharing mechanism is usually modeled by a graph. For simplicity in the early stages of consensus algorithm development, the communication graph is assumed to be fixed. However, a consensus algorithm that is robust or adaptive to topology variations is more desirable, since many practical conditions can be modeled as time-varying communications, for example asynchronous updating, or communication link failures and creations. As communication among agents is an important topic in the MAS literature, various communication assumptions and consensus results have been investigated by researchers (Moreau, 2005; Hatano and Mesbahi, 2005; Tahbaz-Salehi and Jadbabaie, 2008; Zhang and Tian, 2009). An excellent survey can be found in Fang and Antsaklis (2006). Since graph theory is seldom used in control theory and applications, a brief introduction to the topic is given in Appendix A.

A consensus algorithm is a very simple local coordination rule which can result in very complex and useful behaviors at the group level. For instance, it is widely observed that by adopting such a strategy, a school of fish can improve the chance of survival under the sea (Moyle and Cech, 2003). Many interesting coordination problems have been formulated and solved under the framework of consensus, for example distributed sensor fusion (Olfati-Saber *et al.*, 2007), satellite alignment (Ren and Beard, 2008), multi-agent formation (Ren *et al.*, 2007), synchronization of coupled oscillators (Ren, 2008b), and optimal dispatch in power systems (Yang *et al.*, 2013). The consensus problem is usually studied in the infinite time horizon, that is, the consensus is reached as time tends to infinity. However, some finite-time convergence algorithms are available (Cortex, 2006; Wang and Hong, 2008; Khoo *et al.*, 2009; Wang and Xiao, 2010; Li *et al.*, 2011). In the existing literature, most consensus algorithms are model based. By incorporating ILC into consensus algorithms, the prior information requirement from a plant model can be significantly reduced. This advantage will be shown throughout this book.

1.3 Motivation and Overview

In practice, there are many tasks requiring both repetitive executions and coordination among several independent entities. For example, it is useful for a group of satellites to orbit the earth in formation for positioning or monitoring purposes (Ahn *et al.*, 2010). Each satellite orbiting the earth is a repeated task, and the formation task fits perfectly in the ILC framework. Another example is the cooperative transportation of a heavy load by multiple mobile robots (Bai and Wen, 2010; Yufka *et al.*, 2010). In such kinds of task implementation, the robots have to maneuver in formation from the very beginning to the destination. The economic dispatch problem in power systems (Xu and Yang, 2013; Yang *et al.*, 2013) and formation control for ground vehicles with nonholonomic constraints (Xu *et al.*, 2011) also fall in this category. These observations motivate the study of multi-agent coordination control from the perspective of ILC.

As discussed in the previous subsection, the consensus tracking problem is an important multi-agent coordination problem, and many other coordination problems can be formulated and solved in this framework, such as the formation, cooperative search, area coverage, and synchronization problems. We chose consensus tracking as the main topic in this book. Here we briefly describe a prototype consensus tracking problem and illustrate the concepts of distributed tracking error which are used throughout the book. In the problem formulation, there is a single leader that follows a prescribed trajectory, and the leader's behavior is not affected by others in the network. There are many followers, and they can communicate with each other and with the leader agent. However, they may not know which one the leader is. Due to communication limitations, a follower is only able to communicate with its near neighbors. The control task is to design an appropriate *local* controller such that all the followers can track the leader's trajectory. A local controller means that an agent is only allowed to utilize local information. To illustrate these concepts, Figure 1.2 shows an example of a communication network. (Please see Appendix A for a revision of graph theory.) Each node in the graph represents an agent (agents will be modeled by dynamic systems in later chapters). Edges in the graph show the information flow. For instance, there is an edge starting from agent 2 and ending at agent 1, which means agent 1 is able to obtain information from agent 2. In this example there are two edges ending at agent 1. This implies that agent 1 can utilize the information received from agents 0 and 2. Let x_i denote the variable of interest for the ith agent, for instance, position, velocity, orientation, temperature, pressure, and so on. The distributed error ξ_1 for agent 1 is defined as

$$\xi_1 = (x_0 - x_1) + (x_2 - x_1).$$

The distributed error ξ_1 will be used to construct the distributed learning rule.

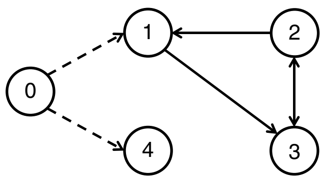

Figure 1.2 Example of a network.

With this problem in mind, the main content of the book is summarized below.

1) In Chapter 2, a general consensus tracking problem is formulated for a group of global Lipschitz continuous systems. It is assumed that the communication is fixed and connected, and the perfect identical initialization condition (*iic*) constraint is satisfied as well. A D-type ILC rule is proposed for the systems to achieve perfect consensus tracking. By adoption of a graph dependent matrix norm, a local convergence condition is devised at the agent level. In addition, optimal learning gain design methods are developed for both directed and undirected graphs such that the λ-norm of tracking error converges at the fastest rate.
2) In Chapter 3, we investigate the robustness of the D-type learning rule against communication variations. It turns out that the controller is insensitive to iteration-varying topology. In the most general case, the learning controller is still convergent when the communication topology is uniformly strongly connected over the iteration domain.
3) In Chapter 4, the PD-type learning rule is proposed to deal with imperfect initialization conditions as it is difficult to ensure perfect initial conditions for all agents due to sparse information communication—hence only a few of the follower agents know the desired initial state. The new learning rule offers two main features. On the one hand, it can ensure controller convergence. On the other hand, the learning gain can be used to tune the final tracking performance.
4) In Chapter 5, a novel input sharing learning controller is developed. In the existing literature, when designing the learning controller, only the tracking error is incorporated in the control signal generation. However, if the follower agents can share their experience gained during the process, this may accelerate the learning speed. Using this idea, the new controller is developed for each agent by sharing its learned control input with its neighbors.
5) In Chapter 6, we apply the learning controller to a formation problem. The formation contains two geometric configurations. The two configurations are related by a high-order internal model (HOIM). Usually the ILC control task is fixed. The most challenging part of this class of problem is how to handle changes in configuration. By incorporating the HOIM into the learning controller, it is shown that, surprisingly, the agents are still able to learn from different tasks.
6) In Chapter 7, by combining the Lyapunov analysis method and contraction-mapping analysis, we explore the applicability of the P-type learning rule to several classes of local Lipschitz systems. Several sufficient convergence conditions in terms of Lyapunov criteria are derived. In particular, the P-type learning rule can be applied to a Lyapunov stable system with quadratic Lyapunov functions, an exponentially stable system, a system with bounded drift terms, and a uniformly bounded energy bounded state system under control saturation. The results greatly complement the existing literature. By using the results of this chapter, we can immediately extend the results in Chapters 2–5 to more general nonlinear systems.
7) In Chapter 8, the composite energy function method is utilized to design an adaptive learning rule to deal with local Lipschitz systems that can be modeled by system dynamics that are linear in parameters. With the help of a special parameterization method, the leader's trajectory can be treated as an iteration-invariant parameter

that all the followers can learn from local measurements. In addition, the initial rectifying action is applied to reduce the effect of imperfect initialization conditions. The method works for high-order systems as well.

8) Chapter 9 addresses the consensus problem of nonlinear multi-agent system (MAS) with state constraints. A novel type of barrier Lyapunov function (BLF) is adopted to deal with the bounded constraints. An ILC strategy is introduced to estimate the unknown parameter and basic control signal. To address the consensus problem comprehensively from both theoretical and practical viewpoints, five control schemes are designed in turn: the original adaptive scheme, a projection-based scheme, a smooth function based scheme as well as its alternative, and a dead-zone like scheme. The consensus convergence and constraints guarantee are strictly proved for each control scheme by using the barrier composite energy function (BCEF) approach.
9) Lagrangian systems have wide applications in practice. For example, industry robotic manipulators can be modeled as a Lagrangian system. In Chapter 10, we develop a set of distributed learning rules to synchronize networked Lagrangian systems. In the controller design, we fully utilize the inherent features of Lagrangian systems, and the controller works under a directed acyclic graph.
10) In Chapter 11, we focus our attention on discrete-time system and present a generalized iterative learning algorithm to solve an optimal power generation problem in a smart grid. Usually the optimal power dispatch problem is solved by centralized methods. Noticing that the optimal solution is achieved when the incremental costs for all power generators are equal, if we consider the incremental cost as the variable of interest, it may be possible to devise a distributed algorithm. Following this idea and by virtue of the distributed nature of the consensus algorithm, a hierarchical two-level algorithm is developed. The new learning algorithm is able to find the optimal solution, as well as taking power generator constraints and power line loss into account.

1.4 Common Notations in This Book

The set of real numbers is denoted by $\mathbb{R}$, and the set of complex numbers is denoted by $\mathbb{C}$. The set of natural numbers is denoted by $\mathbb{N}$, and $i \in \mathbb{N}$ is the number of iteration. For any $z \in \mathbb{C}$, $\Re(z)$ denotes its real part. For a given vector $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^n$, $|\mathbf{x}|$ denotes any l_p vector norm, where $1 \leq p \leq \infty$. In particular, $|\mathbf{x}|_1 = \sum_{k=1}^{n} |x_k|$, $|\mathbf{x}|_2 = \sqrt{\mathbf{x}^T\mathbf{x}}$, and $|\mathbf{x}|_\infty = \max_{k=1,\ldots,n} |x_k|$. For any matrix $A \in \mathbb{R}^{n\times n}$, $|A|$ is the induced matrix norm. $\rho(A)$ is its spectral radius. Moreover, $\otimes$ denotes the Kronecker product, and I_m is the $m \times m$ identity matrix.

Let $C^m[0, T]$ denote a set consisting of all functions whose mth derivatives are continuous on the finite-time interval $[0, T]$. For any function $\mathbf{f}(\cdot) \in C[0, T]$, the supremum norm is defined as $\|\mathbf{f}\| = \sup_{t\in[0,T]} |\mathbf{f}(t)|$. Let λ be a positive constant, the time weighted norm (λ-norm) is defined as $\|\mathbf{f}\|_\lambda = \sup_{t\in[0,T]} e^{-\lambda t}|\mathbf{f}(t)|$.