ITERATIVE LEARNING CONTROL AND ITS APPLICATIONS

PENGHAI ZHAO

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF SCIENCE

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

JANUARY 2021

Abstract

Robotic manipulators and Unmanned Aerial Vehicles (UAVs) have been used to execute some repeatable assignments, due to the advantage of safety, convenience and flexibility. Iterative learning control (ILC) is an approach to eliminate some repeatable disturbance which may come from unknown parameters, dynamic uncertainties or the surroundings. Therefore, this research aims to present two types of iterative learning controllers, PD-type and adaptive-type, to implement on robotic manipulators and UAVs, which would complete the given repetitive missions and achieve the expected specifications. PD-type and adaptive-type ILC are tested on SRV02 equipment with a rigid manipulator, to eliminate the repetitive unknown disturbance. Meanwhile, a dead zone inverse model is proposed to solve the actuator dead zone problem, which is verified using the same equipment with a flexible manipulator. Compares with PD-type test on manipulator, adaptive-type ILC is decided to implement on UAVs in this research. The traditional hierarchical control

method for UAVs is adopted. Attitude and position control systems are designed based on the same adaptive-type ILC algorithm, however, the experimental test are finished separately. The inner loop control performance are verified using Gimbal which is a frame that UAVs can be setup on it with two degrees of freedom, including roll and yaw angles. The free flight experimental test is completed with the purpose of certifying the proposed out loop control strategy. In addition, theoretical proof and simulation results are also presented to demonstrate the effectiveness of the proposed controllers.

Dedication

I dedicate this thesis to people I love who love me in return

Acknowledgements

I would foremost like to acknowledge, with gratitude, Professor Jinjun Shan for his supervision. His intelligence and wisdom have greatly enhanced and enlightened my inexperienced skills and knowledge in research. His advice and guidance was invaluable on completing this work. Without his generous educational and financial support, this research would have been nowhere near as it is.

I would like to thank Dr. Ti Chen for his invaluable guideline and advice throughout my graduate study. Sincere thanks to my lab mates Dr. Shiyuan Jia, Dr. Zeng Wang, Dr. Yuying Liang, Marc Savoie, Hassan Alkomy, Samira Eshghi, Mingfeng Yuan, and Yibo Liu, for various help they have offered me.

Last, but certainly not least, I would like to thank my family and friends, my room mates, Caoyi Chen and Shifei Yang, and Shihan Gong, Yutong Zhang and Penghan Wang for their mental support.

Table of Contents

List of Tables

List of Figures

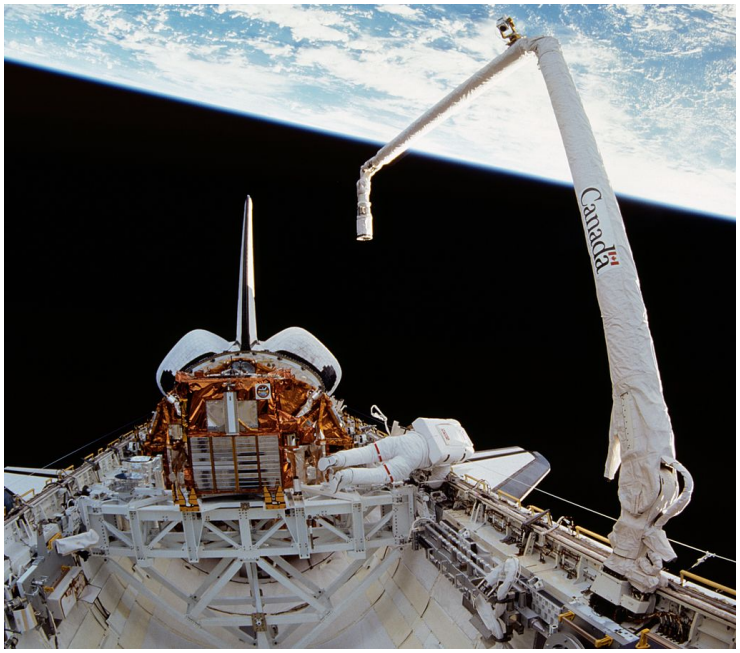
1 Introduction

1.1 Motivation

Robotic manipulator is a instrument, an arm-like mechanism which includes a series of parts that can be moved with a number of degrees of freedom, used to operate materials without direct physical contact by the operator [5]. Initially, they were used in inaccessible places. In recent decades, there has been increased interests in the development of robotic manipulators, including rigid and flexible manipulators [6]. Robotic manipulators are employed to implement in wide range of applications, such as robotic surgery [7], space [8] (Fig. 1.1(a)) and manufacturing tasks [9] (Fig. 1.1), i.e, transporting [10] and assembling [11].

In industrial environment, manipulator usually performs the transporting and assembling missions, which are repetitive. Furthermore, the rigid or flexible manipulators usually operate with uncertain parameters in real engineering. In this case,

there are some unknown repetitive disturbance [12, 13, 14, 15, 16], such as the payload, frictional force, and vibration and dead-zone problems [17, 18, 19, 20, 21, 22]. Many research use different approaches to solve the problem, including neural-network control [23] and sliding-mode control [24]. Generally, the aforementioned literature can eliminate these kinds of disturbance, however, these methods do not take advantage of repetitive missions and characteristic of repetitive disturbance. This research tries to present control algorithms, which can remove the repetitive disturbance and dead-zone based on the characteristic of repetitive missions.



(a) Canadarm during Soace Shuttle Mission STS-72 [1]

(b) robotic manipulator [25]

(c) 6-DOF robotic manipulator arm [26]

Figure 1.1: Example of robotic manipulators

Unmanned Aerial Vehicles (UAVs) have been gained much attention and been the subject of significant study for decades, which can perform operations that could be harmful to humans or that would require more invest of resources if done by humans, it would require fewer resources to use an UAV to check up the condition of machinery, structures or infrastructures located on remote areas [27, 28, 29, 30]. Additionally, UAVs are also appealing for civilian [31, 32] use like aerial photo [33], transport [34, 35], terrain detection [36, 37], deliveries [38, 39] and even data collection [40, 41].

Figure 1.2: Food drone delivery [2]

Quadrotors use varying rotor speeds to maneuver [42], which are widely used in aerial tasks, since the capability of vertical take-off and land, high agility [43]. The dynamic model of quadrotors is nonlinear and under-actuated, since there are less actuators than the degrees of freedom. Similarly, the quadrotors can execute the repetitive missions [44], and most of research relied on complicated dynamic model

which causes intense computation. Hence, this research is focusing on the linearized dynamic model of quadrotors, and using the same control algorithms as in robotic manipulators to track repetitive trajectories with unknown repetitive disturbance.

Iterative learning control (ILC) algorithms are designed to improve the present control performance of a system by learning from the past perform experience, which is widely used in industrial applications [45, 46]. The advantage of ILC algorithms is dealing with tasks performed repeatedly. They have the robustness to eliminate the system uncertainties and unknown disturbance and the simplicities to apply a system. Motivated by the aforementioned superiority of ILC algorithms, this research is focusing on apply ILC algorithms to robotic manipulators and UAVs to perform repetitive missions, respectively.

1.2 Objectives and Methodology

The goal of this research is to take superiority of ILC algorithms and use two types of ILC algorithms, PD-type ILC and adaptive-type ILC, to present the learning control systems for both robotic manipulators [47, 48, 49] and UAVs [50, 51, 52] to improve the trajectory tracking performance.

Usually, for the robotic manipulators, when driven it to complete some repetitive tasks, there are some unknown disturbance. This research aims to use the present

ILC controllers to compensate the effect of dead-zone and some unknown disturbance. Dead-zone is a phenomenon that happens during an interval where the output of the control system is zero. For robotic manipulators, it happens during the manipulator changes the rotation direction, such as from clockwise to counter clockwise. To remove the effect of it, this research will add a dead-zone inverse, which is designed to convert the auxiliary control input to control input. Then this research will compare these two ILC controllers to determine the advantages and disadvantages. In order to verify the theoretical analysis, Simulink model is used, and the experimental test is finished by using Quanser SRV02 equipment which is from Quanser company. The robotic manipulator is assembled on SRV02 equipment.

The UAV equipment is also from Quanser company which is called QDrone. First, implement the ILC controllers, which are verified on robotic manipulator, on the UAV attached to the Gimbal to track the desired repetitive time-varying attitude. After that, implement the ILC controllers on UAV to track the desired repetitive time-varying trajectory on different directions, respectively, as well as focusing on the performance of the position and the attitude stabilization at the same time. Meanwhile, as the number of iteration increases, the tracking performance could be improved.

The simulation of the previously mentioned objectives can be achieved using Matlab & Simulink. The test of UAVs can be finished in York University Autonomous Unmanned Vehicle (YU-UAV) facility in Spacecraft Dynamics Control and Navigation Laboratory (SDCNLab). There are 5 markers on the top of the QDrone with a unique shape, and there are 16 cameras on the wall to locate the position of the QDrone.

1.3 Organization of Thesis

The contents in this thesis are organized as follows:

Chapter 2: Iterative Learning Control - to provide basic information about iterative learning control method including its history, advantages and two types of iterative learning control algorithms which are used in this research.

Chapter 3: Application of Iterative Learning Control (ILC) to Robotic Manipulators - to present the dynamic model of both rigid and flexible manipulator, and the implementation of ILCs to them, finally, the correlative simulation and experimental results are attached.

Chapter 4: Application of Iterative Learning Control (ILC) to UAVs - to describe the dynamic model of a UAV, including the general translation of a UAV, and

provide the controller design based on attitude and position separately. To demonstrate the performance, simulation and experimental results are provided.

Chapter 5: Conclusion and Future Work - to provide general reviews of this thesis and to have a discussion on the possible future development.

2 Iterative Learning Control

This chapter is devoted to describe and summarized past studies and some investigation in Iterative learning control method. Firstly, a brief introduction of ILC is proposed in Section 2.1. Two different types of ILC algorithms for two types of systems are shown in Section 2.2 and 2.3, respectively. Finally, conclusion of this chapter are summarized in Section 2.4.

2.1 Introduction of Iterative Learning Control (ILC)

Iterative learning control is different from most other control methods, because this algorithm uses the past control experience, such as control input signals and tracking errors, to improve the current control performance [53]. It is same as the human learning behavior by repeating or practicing same tasks for many times, the performance of human behavior would be better as one learns each time. In this case, ILC is widely used in systems that operate some specific tasks, repeatedly,

such as robot manipulators for transporting and assembling, quadrotors for data collection and delivery, etc. There are two phases in iterative learning control: first the long term memory components are used to store past control information, then the stored control information is fused in certain manner so as to ensure that the system meets control specifications such as convergence, robustness, etc [54]. ILC was first proposed by [55] in 1984. It should be noted that, ILC does not require full information about the dynamic model of the system to generate the desired dynamic behaviors. Due to the mentioned advantages, many researchers paid much attention to it [56].

In the past decades, researchers found that the combination of ILC and other control techniques may generate better controllers that meet the desired control performance which is impossible for any individual approach. Integration of classical PD, adaptive, newton-method or other learning algorithms into ILC to eliminate various control problems has been reported. Since both robotic manipulators and quadrotors are considered to operate some repetitive missions, ILC is considered to be the control algorithm in this research. As mentioned before, there are various types of ILC algorithms. This research selects two types as the desired controller, which are PD-type ILC and adaptive-type ILC.

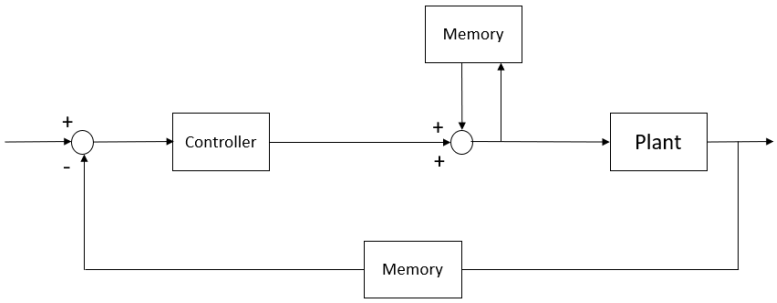


Figure 2.1: Structure of iterative learning control [57]

2.2 A Simplest ILC Example

A simplest ILC example is proposed here to describe the fundamental principles and concepts of ILC [57]. For a known process,

$$y(t) = g(t)u(t) \tag{2.1}$$

where $g(t) \neq 0$ is bounded defined over a period $[0,\ T]$, the objective is to get the control input, $u(t)$, which ensures that the desired bounded trajectory $y_d(t) \forall t \in [0,\ T]$ can be tracked and the tracking error $e(t)$ decreases.

If $g(t)$ is given, so that the desired control input can be calculated directly by inverting the process, which is an open-loop approach

$$u_d(t) = \frac{y_d(t)}{g(t)} \quad \forall t \in [0,\ T] \tag{2.2}$$

However, the open-loop control scheme is sensitive to the plant modeling error and

any other uncertainties. So that assume $g(t)$ is unknown and bounded with

$$0 < \alpha_1 \leq g(t) \leq \alpha_2 < \infty \tag{2.3}$$

where α_1 and α_2 are known lower and upper bounds. When the the desired trajectory $y_d(t)$ is repeated, the control input can be calculated iteratively by the following simple iterative learning control algorithm,

$$u_{i+1} = u_i + pe_i(t) \quad \forall t \in [0,\ T] \tag{2.4}$$

where i presents the ith iteration, $u_0(t)$ is set zero. p is a positive constant learning gain, and $e_i(t) = y_d(t) - y_i(t)$ is the defined tracking error. It should be noted that, once the desired trajectory $y_d(t)$ is given, the desired control input $u_d(t)$ is fixed. In order to prove the convergence of the proposed ILC algorithm, as $i \to \infty$, $e_i(t) \to 0$ or $\Delta u_i(t) = u_d(t) - u_i(t) \to 0$, there are two methods.

Method 1. Output Tracking

$$e_{i+1} = y_d - y_{i+1} = y_d - g(u_i + pe_i) = (1 - pg)e_i \tag{2.5}$$

So that

$$\mid e_{i+1} \mid \leq \mid 1 - pg \mid\mid e_i \mid \tag{2.6}$$

Method 2. Control Input

$$\Delta u_{i+1} = u_d - u_{i+1} = u_d - (u_i + pe_i) = \Delta u_i - pe_i \tag{2.7}$$

And

$$e_i = y_d - y_i = gu_d - gu_i = g\Delta u_i \tag{2.8}$$

By substituting e_i into Eq. (2.7)

$$\Delta u_{i+1} = (1 - pg)\Delta u_i \tag{2.9}$$

So that

$$\mid \Delta u_{i+1} \mid \leq \mid 1 - pg \mid\mid \Delta u_i \mid \tag{2.10}$$

The above two proof approaches illustrate the proposed ILC algorithm works, which makes the convergence of u_i to u_d (or y_i to y_d) as $i \to \infty$. And describes the basic information about ILC. However, for more complicated situations such as multi-input and multi-output (MIMO) case, dynamic models or some uncertainties, more complex controller should be applied.

2.3 PD-Type Iterative Learning Control (ILC)

PD control is a classical control method, which is widely used in applications, due to its simple structure and easy to tune the control gains. The integration of PD control and ILC control remains the advantages of both algorithm [54].

2.3.1 Problem Formulation

Consider the linear time invariant system described by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2.11}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^l$ and $y(t) \in \mathbb{R}^r$ denote the state, input and output, respectively. A, B and C are matrices with appropriate dimensions and it is assumed that CB is nonsingular.

Denote $x_d(t)$ is the desired state trajectory which is continuously differentiable on $[0, T]$. The objective is to find the desired control input $u_d(t)$, which makes the tracking error $e_i(t) = y_d(t) - y_i(t)$ converges to zero, where i presents the ith-iteration. The following PD-type iterative learning control algorithm is adopted,

$$u_{i+1}(t) = u_i + \Gamma(\dot{e}_i(t) - Re_i(t)) \tag{2.12}$$

The initial condition at each iteration remains the same, i.e., $x_i(0) = x_0$.

2.3.2 Convergence of PD-Type ILC in the Sense of Sup-norm

Theorem Suppose that the PD-type ILC control law is applied to the system with Γ such that the convergence condition holds, that is $\rho = \| I - \Gamma CB \|_\infty < 1$. Also assume that the initial error is zero at each iteration. If the desired trajectory

is given on the interval $t \in [0, T_{sup}]$ where T_{sup} is bounded as

$$T_{sup} < \frac{1}{a} \ln(1 + \frac{a(1-\rho)}{hb}) \tag{2.13}$$

where

$$a = \| A \|_\infty, \quad b = \| B \|_\infty, \quad h = \| \Gamma(CA - RC) \|_\infty \tag{2.14}$$

then, there exists $\rho_0 < 1$ such that

$$\| e_{i+1}(t) \|_m \leq \rho_0 \| e_i(t) \|_m \tag{2.15}$$

where

$$\| e_i(t) \|_m = \max_{t \in [0, T_{sup}]} \| e_i(t) \|_\infty \tag{2.16}$$

Proof. From the control law and system,

$$e_{i+1}(t) = (I - \Gamma CB)e_i(t) - \Gamma(CA - RC) \int_0^t e^{A(t-\tau)} B e_i(\tau) d\tau \tag{2.17}$$

By taking the $\| \cdot \|_\infty$-norm on both side,

$$\| e_{i+1}(t) \|_\infty = \rho \| e_i(t) \|_\infty + \| \Gamma(CA - RC) \int_0^t e^{A(t-\tau)} B e_i(\tau) d\tau \|_\infty \tag{2.18}$$

so that,

$$\begin{aligned} \| e_{i+1}(t) \|_m &\leq \rho \| e_i(t) \|_m + h \max_{t \in [0, T_{sup}]} \| \int_0^t e^{A(t-\tau)} d\tau \|_\infty \| B \|_\infty \| e_i(t) \|_m \\ &\leq \rho \| e_i(t) \|_m + h \max_{t \in [0, T_{sup}]} \int_0^t e^{a(t-\tau)} d\tau b \| e_i(t) \|_m \\ &= \left(\rho + \frac{1}{a} hb(e^{aT_{sup}} - 1) \right) \| e_i(t) \|_m \end{aligned} \tag{2.19}$$

From the above equation, the exponential convergence will be guaranteed with respect to the sup-norm if

$$\rho_0 = \rho + \frac{1}{a} hb(e^{aT_{sup}} - 1) < 1 \tag{2.20}$$

where

$$\begin{aligned} \rho_0 &= \rho + \frac{1}{a} hb(e^{aT_{sup}} - 1) \\ &< \rho + \frac{1}{a} hb(e^{a\frac{1}{a}\ln(1+\frac{a(1-\rho)}{hb}} - 1) \\ &= \rho + \frac{1}{a} hb(1 + \frac{a(1-\rho)}{hb} - 1) \\ &= 1 \end{aligned} \tag{2.21}$$

If T_{sup} is bounded by assumption, so that the condition holds. This completes the proof.

2.4 Adaptive-Type Iterative Learning Control (ILC)

Adaptive control [58, 59, 60, 61, 62] is a control method which is used to adapt the parameters of the system, the parameters can be unknown or vary. In this section, an adaptive-type ILC is shown for robotic systems [54].

2.4.1 Problem Formulation

Consider a robot system with n rigid links, the mathematical model of which is

$$D(q(t))\ddot{q}(t) + B(q(t), \dot{q}(t))\dot{q}(t) + F(q(t), \dot{q}(t)) + T_a(t) = T(t) \tag{2.22}$$

where $q(t) \in \mathbb{R}^n$ is the generalized joint coordinate vector, $D(q(t)) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $B(q(t), \dot{q}(t)) \in \mathbb{R}^n$ is the centripetal plus Coriolis force matrix, $F(q(t), \dot{q}(t)) \in \mathbb{R}^n$ is the gravitational plus frictional forces, $T(t) \in \mathbb{R}^n$ is the joint torque vector, and $T_a(t) \in \mathbb{R}^n$ is the unknown disturbance vector which is assumed to be bounded and periodic. The symmetric inertia matrix $D(q(t)) \in \mathbb{R}^{n\times n}$ is assumed to be positive definite and bounded as

$$0 \leq \lambda_1 I \leq D(q(t)) \leq \lambda_2 I \quad \forall t \in [0, t_f] \tag{2.23}$$

where $\lambda_1, \lambda_2 > 0$ and I is an $n \times n$ identity matrix. The matrix $\dot{D}(q(t)) - 2B(q(t), \dot{q}(t))$ is assumed to be skew-symmetric as

$$z^T(\dot{D} - 2B)z = 0 \quad \forall z \in \mathbb{R}^n \text{ and } z \neq 0 \tag{2.24}$$

When the desired trajectory $q_d(t) \in \mathbb{R}^n$ is specified as a reference input for system Eq. (2.22), the fundamental control problem is to find a control input $T(t)$ with which the system output $q(t)$ follows $q_d(t)$ $\forall t \in [0, t_f]$ as close as possible. In the framework of learning control, this objective can be stated as follows:

Problem Statement. Suppose that $q_d(t) \in [0, t_f]$, the trajectory vector of system Eq. (2.22), is in the interior of a domain Q which is a closed, bounded, and simply connected subset of $\mathbb{R}^n$. Then, find a sequence of piecewise continuous control input $T^j(t) \in \mathbb{R}^n$ ($t \in [0, t_f]$) for uncertain system Eq. (2.22) with which the system trajectory $q^i(t)$ follows $q_d(t)$ with a given accuracy ϵ. In other words for a given $\epsilon > 0$, there exists a positive integer N such that

$$| \, q_d(t) - q^j(t) \, | \leq \epsilon, \quad \forall t \in [0, t_f], \; j \geq N \tag{2.25}$$

where j denotes the jth-iteration. This means that q^j converges uniformly to q_d.

In the following, the uncertain system Eq. (2.22) is assumed to be repetitive for all $t \in [0, t_f]$ and the operating conditions such as sampling frequency, payloads scheduling, and initial configuration etc. are all assumed to be prespecified. The desired joint position, velocity, acceleration and control input vectors are denoted as $q_d(t)$, $\dot{q}_d(t)$, $\ddot{q}_d(t)$ and $T_d(t)$, respectively, and the actual joint position, velocity, acceleration and control input vectors at the jth iteration are denoted as $q^j(t)$, $\dot{q}^j(t)$, $\ddot{q}^j(t)$ and $T^j(t)$, respectively. For notational brevity, the time argument t will be omitted in the sequel.

2.4.2 Learning Control without Parameter Adaptation

In the learning control scheme, the control input T^j for the jth iteration consists of three input components:

$$T^j = E^j + C^j + H^j \tag{2.26}$$

where E_i is the feedback control input, C^j is the nonlinear compensation term that generates computed torque error, and H^j is the learning control input. The feedback control input E^j is computed from the conventional proportional plus derivative (PD) type control algorithm

$$E^j = \beta L(\dot{e}^j + \alpha e^j) \tag{2.27}$$

where $e^j = q_d - q^j$, β is a positive constant, L is a symmetric positive definite matrix and α is a positive scale factor. The nonlinear compensation term C^j compensates for the nonlinear part of the robotic system and helps keep the feedback gain of E^j reasonably small. The learning control input H^j drives the system to track the reference trajectory over the sequence of iterations and converges to the desired input trajectory function $T_d(t)$. When the system parameters of Eq. (2.22) are completely known, the nonlinear compensation term C^j is of the form

$$C^j = D_e(q^j)\ddot{q}_d + B_e(q^j, \dot{q}^j)\dot{q}_d + F_e(q^j, \dot{q}^j) + \alpha(D(q^j)\dot{e}^j + B(q^j, \dot{q}^j)e^j) \tag{2.28}$$

where $D_e(q^j) = D(q^j) - D(q_d)$, $B_e(q^j, \dot{q}^j) = B(q^j, \dot{q}^j) - B(q_d, \dot{q}_d)$ and $F_e(q^j, \dot{q}^j) = F(q^j, \dot{q}^j) - F(q_d, \dot{q}_d)$.

Substituting the torque input Eq. (2.26) into Eq. (2.22), and applying Eq. (2.27) and Eq. (2.28), it can be obtained that

$$D(q^j)\dot{z}^j + B(q^j, \dot{q}^j)z^j + \beta L z^j = T_d - H^j = \tilde{U}^j \tag{2.29}$$

where $z^j = \dot{e}^j + \alpha e^j$, $T_d = D(q_d)\ddot{q}_d + B(q_d, \dot{q}_d) + F(q_d, \dot{q}_d) + T_a$.For the subsequent development of the learning controller, the following assumption is listed

Assumption The desired control input T_d is piecewise continuous on $[0, t_f]$ and each element T_i of T_d is bounded with known bound T_i^b. This means that $\mid T_i \mid \leq T_i^b$ for all $i = 1, 2, 3, \cdots n$, where n is the number of elements of T_d.

As a motivation to generate a learning algorithm, a Lyapunov function candidate $W(z^j)$ is defined as $\frac{1}{2}{z^j}^T D(q^j)z^j$. Then the derivative of $W(z^j)$ along the error trajectory is

$$\dot{W}(z^j) = {z^j}^T D(q^j)\dot{z}^j + \frac{1}{2}{z^j}^T \dot{D}(q^j)z^j = -\beta {z^j}^T L z^j + {z^j}^T \tilde{U}^j \tag{2.30}$$

Integrating both sides of the above equation

$$W(z^j(t)) - W(z^j(0)) = -\int_0^t \beta {z^j}^T L z^j d\tau + \int_0^t {z^j}^T \tilde{U}^j d\tau \tag{2.31}$$

Therefore, the learning algorithm is proposed as

$$H^{j+1} = Proj\{\bar{H}^{j+1}\} = \{Proj\{\bar{H}_1^{j+1}\}, ..., Proj\{\bar{H}_n^{j+1}\}\} \tag{2.32}$$

where $\bar{H}^{j+1} = H^j + \beta L z^j$ and

$$Proj\{\bar{H}_i^{j+1}\} = \begin{cases} T_i^b & \text{if} \quad \bar{H}_i^{i+1} \geq T_i^b \\ -T_i^b & \text{if} \quad \bar{H}_i^{j+1} \leq -T_i^b \\ \bar{H}_i^{j+1} & \text{otherwise} \end{cases} \tag{2.33}$$

As an initial condition, $z^j(0)$ is set to 0 (i.e., $e^j(0) = 0$ and $\dot{e}^j(0) = 0$).

Theroem The control law with the learning rule converges as

(1) $\lim_{j\to\infty} V^j(t) = V(t)$

(2) $\lim_{j\to\infty} z^j(t) = 0$, for all $t \in [0, t_f]$

where V^j is the performance index function

$$V^j(t) = \int_0^t \tilde{U}^{j^T}(\tau) L^{-1} \tilde{U}^j(\tau) d\tau \tag{2.34}$$

Proof. From the definition of $\tilde{U}^j$ and the learning rule

$$\Delta\tilde{U}^j = \tilde{\tilde{U}}^{j+1} - \tilde{U}^j = -\beta L z^j \tag{2.35}$$

where $\tilde{\tilde{U}}^j = T_d - \bar{H}_i$. Since $\mid \tilde{U}^j \mid \leq \mid \tilde{\tilde{U}}^j \mid$

$$V^{j+1} - V^j \leq \bar{V}^{j+1} - V^j \tag{2.36}$$

where $\bar{V}^j = \int_0^t \tilde{\bar{U}}^{j^T}(\tau) L^{-1} \tilde{\bar{U}}^j(\tau) d\tau$,

$$
\begin{aligned}
V^{j+1} - V^j &\leq \bar{V}^{j+1} - V^j \\
&= \int_0^t (\tilde{\bar{U}}^{j+1^T} L^{-1} \tilde{\bar{U}}^{j+1} - \tilde{U}^{j^T} L^{-1} \tilde{U}^j) d\tau \\
&= \int_0^t (\Delta \tilde{U}^{j^T} L^{-1} \tilde{U}^j + 2\Delta \tilde{U}^{j^T} L^{-1} \tilde{U}^j) d\tau \\
&= \int_0^t (\beta^2 z^{j^T} L z^j - 2\beta z^{j^T} (D(q^j)\dot{z}^j + B(q^j, \dot{q}^j) z^j + \beta L z^j)) d\tau \\
&= -2\beta \int_0^t z^{j^T} (D(q^j)\dot{z}^j + B(q^j, \dot{q}^j) z^j) d\tau - \beta^2 \int_0^t z^{j^T} L z^j d\tau \\
&= -\beta z^{j^T} D(q^j) z^j + \beta \int_0^t z^{j^T} (\dot{D}(q^j) - 2B(q^j, \dot{q}^j)) z^j d\tau - \beta^2 \int_0^t z^{j^T} L z^j d\tau \\
&= -\beta z^{j^T} D(q^j) z^j - \beta^2 \int_0^t z^{j^T} L z^j d\tau \\
&\leq -\beta z^{j^T} D(q^j) z^j
\end{aligned} \tag{2.37}
$$

Since V^j is positive definite and monotonically decreasing, V^j converges to some function V [63]. Hence, $V^{j+1} - V^j$ converges to zero. Therefore, z^j converges to zero.

2.4.3 Adaptive Learning Control

When the system parameters are not completely known, then rearrange the dynamic Eq. (2.22) in terms of a set of system parameters, the following algebraic description of the system is obtained,

$$
Y(q(t), \dot{q}(t), \ddot{q}(t))\Theta = T(t) - T_a(t) \tag{2.38}
$$

where $Y(q(t), \dot{q}(t), \ddot{q}(t)) \in \mathbb{R}^{n \times l}$ is the regression matrix and $\Theta \in \mathbb{R}^l$ is a suitably chosen parameter vector. The following assumption is proposed

Assumption Each element Θ_i of the parameter vector Θ is bounded with known bound Θ_i^b. This means that $\mid \Theta_i \mid \leq \Theta_i^b$ for all $i = 1, 2, \cdots, l$, where l is the number of elements of Θ.

Compared with the previously developed learning controller without parameter adaptation, the current learning controller system uses the feedback input

$$E^j = \beta Y_e^j Y_e^{j^T} + \beta L z^j \tag{2.39}$$

The additional term helps to eliminate the constraints that was imposed on L and β in [64]. Next, since the system parameters Θ are not known, the nonlinear compensation input term C^j depends on the estimated parameter vector $\hat{\Theta}^j$ as

$$C^j = \hat{D}_e(q^j)\ddot{q}_d + \hat{B}_e(q^j, \dot{q}^j)\dot{q}_d + \hat{F}_e(q^j, \dot{q}^j) + \alpha(\hat{D}(q^j)\dot{e}^j + \hat{B}(q^j, \dot{q}^j)e^j) \tag{2.40}$$

where $\hat{D}_e(q^j) = \hat{D}(q^j) - \hat{D}(q_d)$, $\hat{B}_e(q^j, \dot{q}^j) = \hat{B}(q^j, \dot{q}^j) - \hat{B}(q_d, \dot{q}_d)$, $\hat{F}_e(q^j, \dot{q}^j) = \hat{F}(q^j, \dot{q}^j) - \hat{F}(q_d, \dot{q}_d)$. Note that $\hat{\cdot}$ denotes the estimated variables. Substituting Eq. (2.39) and Eq. (2.40) into Eq. (2.22), and rearrange the equation

$$D(q^j)\dot{z}^j + B(q^j, \dot{q}^j)z^j + \beta L z^j + \beta Y_e^j Y_e^{j^T} z^j = Y_e^j \tilde{\theta}^j + \tilde{U}^j \tag{2.41}$$

Define $W(z^j)$ as $\frac{1}{2}z^{jT}D(q^j)z^j$,

$$W(z^j(t))-W(z^j(0)) = -\beta\int_0^t \beta z^{jT}(L+Y_e^j Y_e^{jT})z^j d\tau + \int_0^t z^{jT}(\tilde{U}^j + Y_e^j\tilde{\theta}^j)d\tau \quad (2.42)$$

Hence, the parameter learning rule is proposed as

$$\hat{\theta}^{j+1} = Proj\{\hat{\bar{\theta}}^{j+1}\} = \{Proj\{\hat{\bar{\theta}}^{j+1}\}, ..., Proj\{\hat{\bar{\theta}}^{j+1}\}\} \quad (2.43)$$

where $\hat{\bar{\theta}}^{j+1} = \hat{\theta}^j + \beta Y_e^j z^j$ and

$$Proj\{\hat{\bar{\theta}}_i^{j+1}\} = \begin{cases} \theta_i^b & \text{if} \quad \hat{\bar{\theta}}_i^{j+1} \geq \theta_i^b \\ -\theta_i^b & \text{if} \quad \hat{\bar{\theta}}_i^{j+1} \leq -\theta_i^b \\ \hat{\bar{\theta}}_i^{j+1} & \text{otherwise} \end{cases} \quad (2.44)$$

The adaptive-type ILC controller could be:

$$T^j = E^j + H^j + Y_e^j\hat{\Theta}^j \quad (2.45)$$

Theorem The adaptive learning control law for the system converges uniformly as

(1) $\lim_{j\to\infty} V_a^j(t) = V_a(t)$

(2) $\lim_{j\to\infty} z^j(t) = 0$, for all $t \in [0, t_f]$

where

$$V_a^j(t) = \int_0^t (\tilde{U}^{jT}(\tau)L^{-1}\tilde{U}^j(\tau) + \tilde{\Theta}^{jT}(\tau)\tilde{\Theta}^j(\tau))d\tau \quad (2.46)$$

Proof. Since $|\tilde{\tilde{U}}^j \geq| \tilde{U}^j|$ and $|\tilde{\tilde{\Theta}}^j| \geq| \tilde{\Theta}^j|$,

$$V_a^{j+1} - V_a^j \leq \bar{V}_a^{j+1} - \bar{V}_a^j \tag{2.47}$$

where

$$\bar{V}_a^j = \int_0^t (\tilde{\tilde{U}}^{j^T}(\tau) L^{-1} \tilde{\tilde{U}}^j(\tau) + \tilde{\tilde{\Theta}}^{j^T}(\tau) \tilde{\tilde{\Theta}}^j(\tau)) d\tau \tag{2.48}$$

and

$$\tilde{\tilde{\Theta}}^j = \Theta - \hat{\tilde{\Theta}}^j \tag{2.49}$$

$$\Delta\tilde{\Theta}^j = \tilde{\tilde{\theta}}^{j+1} - \tilde{\Theta}^j = \hat{\Theta}^j - \hat{\tilde{\Theta}}^{j+1} = -\beta Y_e^j z^j \tag{2.50}$$

and

$$\begin{aligned} V_a^{j+1} - V_a^j &\leq \bar{V}_a^{j+1} - \bar{V}_a^j \\ &= -2\beta \int_0^t z^{j^T} (D(q^j)\dot{z}^j + B(q^j, \dot{q}^j) z^j) d\tau \\ &\quad -\beta^2 \int_0^t z^{j^T} (L + \beta Y_e^j Y_e^{j^T}) z^j d\tau \end{aligned} \tag{2.51}$$

Integrating the first term by part and exploiting the fact that $\dot{D} - 2B$ is skew-symmetric, the following equation can be obtained,

$$V_a^{j+1}(t) - V_a^j(t) \leq \beta z^{j^T} D(q^j) z^j \tag{2.52}$$

since V^j is positive definite and monotonically decreasing, V^j converges to some function V_a [63]. Then z^j also converges to zero as in the proof of Theorem.

2.5 Conclusions

This chapter illustrates a brief introduction of ILC and the reason why in this research ILC is adopted. And given an simple example to determine the basic principle of ILC. Two types of ILc are shown as the potential method to solve the problems in this research for robotic manipulators and quadrotor.

PD-type ILC is using for the linear time invariant system, which can be described as state-space form. Due to its simple structure and easy to tune the control gains, this method is used for robotic manipulators to operate some specific repetitive missions, like transporting and assembling.

Adaptive-type ILC is using for some more complex systems which can be described as Lagrange equations, including dynamic uncertainties, parameters uncertainties, and etc. This method may also be used into robotic manipulators to verify the control performance and be familiar with it. Additionally, because of the widely use of Lagrange equations, this method is selected as a method to apply on quadrotor to accomplish some tasks which need to repeat many times.

3 Application of ILC to Robotic Manipulators

In this chapter, two kinds of manipulators are introduced, which are rigid manipulator and flexible manipulator. The dynamic model of rigid manipulator and two types of ILC algorithm are illustrated in Section 3.1. Similarly, Section 3.2 presents the model of flexible manipulator based an Euler-Lagrange equation, and describes a new ILC controller to eliminate the effect of uncertain parameters, dead-zone and unknown repetitive disturbance. The simulation and experimental results are provided in Section 3.3 and 3.4, respectively. Finally, concluding remarks of this chapter are summarized in Section 3.5

3.1 Rigid Manipulator

3.1.1 Dynamic Modeling

The dynamic model of one rigid manipulator is considered to be the DC motor model, as shown in Fig. 3.1,

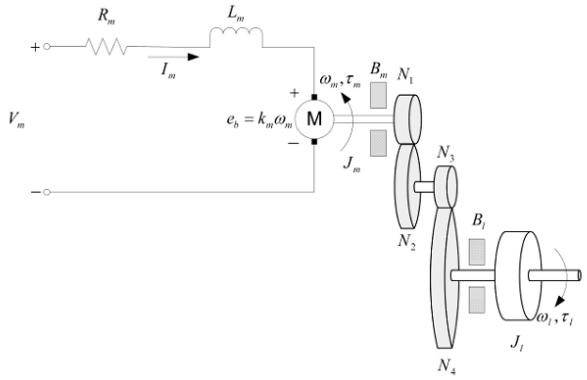


Figure 3.1: SRV02 DC motor armature circuit and gear train [3]

3.1.1.1 Electrical Model

The DC motor system is shown in Fig. 3.1. Denote that R_m is the motor resistance, L_m is the inductance, and k_m is the back-emf constant. The back-emf voltage $e_b(t)$ depends on the motor shaft,w_m, and the back-emf constant of the motor, k_m. It opposes the current flow. The back emf voltage is given by [3]:

$$e_b(t) = k_m w_m(t) \tag{3.1}$$

Using Kirchoff's Voltage Law, the following equation can be obtained,

$$V_m(t) - R_m I_m(t) - L_m \frac{dI_m(t)}{dt} - k_m w_m(t) = 0 \tag{3.2}$$

where $I_m(t)$ is the current cross the circuit. Since the motor inductance L_m is much less than its resistance, it can be ignored. Then, the equation can be rewritten as

$$V_m(t) - R_m I_m(t) - k_m w_m(t) = 0 \tag{3.3}$$

By solving the equation above, the motor current can be found as,

$$I_m(t) = \frac{V_m(t) - k_m w_m(t)}{R_m} \tag{3.4}$$

3.1.1.2 Mechanical Model

In this section the motion speed w_l of the rigid manipulator will be described, with respect to the applied motor torque, τ_m, is developed. According to the Newton's Second Law of Motion and since the SRV02 with rigid manipulator is a one degree-of-freedom rotary system, the follow equation can be obtained,

$$J \cdot \alpha = \tau \tag{3.5}$$

where J is the moment of inertia of the system, α is the angular acceleration of the system, and τ is the sum of the torques being applied to it. As shown in Fig. 3.1, the SRV02 gear train along with the viscous friction acting on the motor shaft, B_m, and the load shaft B_l are considered. So that,

$$J_l \frac{dw_l(t)}{dt} + B_l w_l(t) = \tau_l(t) \tag{3.6}$$

where J_l is the moment of inertia of the load and τ_l is the total torque applied on the load. The load inertia includes the inertia from the gear train and the attached rigid manipulator. The motor shaft equation is expressed as,

$$J_m \frac{dw_m(t)}{dt} + B_m w_m(t) + \tau_{ml}(t) = \tau_m(t) \tag{3.7}$$

where J_m is the motor shaft moment of inertia and τ_{ml} is the resulting torque acting on the motor shaft from the load torque. The torque at the load shaft from an applied motor torque can be written as,

$$\tau_l(t) = \eta_g K_g \tau_{ml}(t) \tag{3.8}$$

where K_g is the gear ratio and η_g is the gearbox efficiency. The planetary gearbox that is directly mounted on the SRV02 motor is represented by the N_1 and N_2 gears in Fig. 3.1. and has a gear ratio of

$$K_{gi} = \frac{N_2}{N_1} \tag{3.9}$$

This is the internal gear box ratio. The motor gear N_3 and the load gear N_4 are directly meshed together and are visible from the outside. These gears comprise the external gear box which has an associated gear ratio of

$$K_{ge} = \frac{N_4}{N_3} \tag{3.10}$$

The gear ratio of the SRV02 gear train is given as

$$K_g = K_{ge} K_{gi} \tag{3.11}$$

Thus, the torque at the motor shaft through the gears can be expressed as,

$$\tau_{ml}(t) = \frac{\tau_l(t)}{\eta_g K_g} \tag{3.12}$$

The motor should shaft K_g times for the output shaft to rotate one revolution.

$$\theta_m(t) = K_g \theta_l(t) \tag{3.13}$$

By taking the time derivative, the relationship between the angular speed of the motor shaft, w_m, and the angular speed of the load shaft, w_l.

$$w_m(t) = K_g w_l(t) \tag{3.14}$$

By substituting, the following equation can be got,

$$J_m K_g \frac{dw_l(t)}{dt} + B_m K_g w_l(t) + \frac{\frac{dw_l(t)}{dt} + B_l w_l(t)}{\eta_g K_g} = \tau_m(t) \tag{3.15}$$

Collecting the coefficients in terms of the load shaft velocity and acceleration gives

$$(\eta_g K_g^2 J_m + J_l)\frac{dw_l(t)}{dt} + (\eta_g K_g^2 B_m + B_l) w_l(t) = \eta_g K_g \tau_m(t) \tag{3.16}$$

Defining the following terms,

$$J_{eq} = \eta_g K_g^2 J_m + J_l \tag{3.17}$$

$$B_{eq} = \eta_g K_g^2 B_m + B_l \tag{3.18}$$

So that, the Eq. (3.16) can be simplified as

$$J_{eq}\frac{dw_l(t)}{dt} + B_{eq} w_l(t) = \eta_g K_g \tau_m(t) \tag{3.19}$$

3.1.1.3 Dynamic Model

The motor torque is proportional to the voltage applied and is described as

$$\tau_m(t) = \eta_m k_t I_m(t) \tag{3.20}$$

where k_t is the current-torque constant, η_m is the motor efficiency. By substituting Eq. (3.4) into Eq. (3.20), here comes the relation among motor torque, input voltage and load shaft speed,

$$\tau_m(t) = \frac{\eta_m k_t (V_m(t) - k_m w_m(t))}{R_m} \tag{3.21}$$

To express this in terms of V_m and w_l, so that,

$$\tau_m(t) = \frac{\eta_m k_t (V_m(t) - k_m K_g w_l(t))}{R_m} \tag{3.22}$$

Substitute above equation into mechanical model,

$$J_{eq}\Big(\frac{d}{dt}w_l(t)\Big) + B_{eq}w_l(t) = \frac{\eta_g K_g \eta_m k_t (V_m(t) - k_m K_g w_l(t))}{R_m} \tag{3.23}$$

By rearranging the above euqation, it can be obtained,

$$\Big(\frac{d}{dt}w_l(t)\Big)J_{eq} + \Big(\frac{k_m \eta_g K_g^2 \eta_m k_t}{R_m} + B_{eq}\Big)w_l(t) = \frac{\eta_g K_g \eta_m k_t V_m(t)}{R_m} \tag{3.24}$$

This equation can be rewritten as,

$$\Big(\frac{d}{dt}w_l(t)\Big)J_{eq} + B_{eq,v}w_l(t) = A_m V_m \tag{3.25}$$

where the equivalent damping term is defined as,

$$B_{eq,v} = \frac{k_m \eta_g K_g^2 \eta_m k_t + B_{eq} R_m}{R_m} \tag{3.26}$$

and the actuator gain is

$$A_m = \frac{\eta_g K_g \eta_m k_t}{R_m} \tag{3.27}$$

3.1.2 Control System Design

3.1.2.1 PD-Type ILC Controller Design

Based on the above section, the dynamic model of a rigid manipulator can be written into State-Space form, which is described as,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{3.28}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^r$ and $y(t) \in \mathbb{R}^r$ denote the position of the manipulator, input voltage and the actual position of the manipulator, respectively. A, B and C are matrices with appropriate dimensions and CB is nonsingular.

Denote $x_d(t)$ be the desired state trajectory which is continuously differentiable on $[0, T]$. The following PD-type learning control law is used,

$$u_{i+1} = u_i + \Gamma(\dot{e}_i(t) - Re_i(t)) \tag{3.29}$$

where $u_i(t)$ and $e_i(t) = y_i(t) - y_d(t)$ are the control input and tracking error between actual output and the desired trajectory, respectively, at the $i - th$ iteration.

Theorem 3.1.1 Suppose that the control law is applied to the system and the initial condition at each iteration remains the same, i.e., $x_i(0) = x_0, i = 0, 1, 2, ...$ If

$$\|I - \Gamma CB\| \leq \rho < 1 \tag{3.30}$$

then,

$$\lim_{k \longrightarrow \infty} y_i = y_d + e^{Rt} C\{x_0 - x_d(0)\} \tag{3.31}$$

uniformly on $t \in [0, T]$.

Proof Let $u_a(t)$ be a control input such that

$$y_d(t) + e^{Rt} C\{x_0 - x_d(0)\} = Ce^{At} x_0 + C \int_0^t e^{A(t-\tau)} B u_a(\tau) d\tau \tag{3.32}$$

define

$$\Delta u_i(t) = u_a(t) - u_i(t) \tag{3.33}$$

so that

$$\Delta u_{i+1}(t) = u_a(t) - u_i(t) - \Gamma(\dot{y}_d(t) - \dot{y}_i(t)) \tag{3.34}$$

$$\Delta u_{i+1}(t) = (I - \Gamma CB) e u_i(t) - \Gamma(CA - RC) \int_0^t e^{A(t-\tau)} B u_i(\tau) d\tau \tag{3.35}$$

Taking the norm $\|\cdot\|_\infty$ on both side of equation and multiplying both side by $e^{-\lambda}t$ and taking the norm $\|\cdot\|_\lambda$, so that

$$\|\Delta u_{i+1}(t)\|_\infty = max_{0\leq t\leq T} e^{-\lambda t}\|\Delta u_{i+1}(t)\|_\infty \tag{3.36}$$

and

$$max_{0\leq t\leq T} e^{-\lambda t}\|\Delta u_{i+1}(t)\|_\infty = (\rho + h\frac{1-e^{(a-\lambda)T}}{\lambda - a})\|\Delta u_i(t)\| \tag{3.37}$$

where $h = \|\Gamma(CA - RC)\Gamma\|_\infty \cdot \|B\|_\infty$, and $a = \|A\|_\infty$. Since $0 \leq \rho < 1$ by assumption, it is possible to choose λ sufficiently large so that

$$\rho_o = \rho + h\frac{1-e^{(a-\lambda)T}}{\lambda - a} < 1 \tag{3.38}$$

Thus,

$$\lim_{i\longrightarrow\infty} \|\Delta u_i(t)\|_\lambda = 0 \tag{3.39}$$

It is clear by definition of the norm $\|\cdot\|_\lambda$ that these convergence are uniform on $t \in [0, T]$. Therefore, $\lim_{k\longrightarrow\infty} u_i(t) = u_a(t)$ uniformly on $[0, T]$. Then,

$$\lim_{i\longrightarrow\infty} y_i = y_d + e^{Rt}C\{x_0 - x_d(0)\} \tag{3.40}$$

uniformly on $t \in [0, T]$. This completes the proof.

3.1.2.2 Adaptive Type ILC Controller Design

Based on section 3.1.1, the dynamic model of the system can be described as followed Lagrange-system,

$$D(q(t))\ddot{q}(t) + B(q(t), \dot{q}(t))\dot{q}(t) + T_a(t) = \tau(t) \tag{3.41}$$

where $q(t) \in \mathbb{R}^n$ is the generalized joint coordinate vector, $D(q(t)) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $B(q(t), \dot{q}(t))q(t) \in \mathbb{R}^n$ is the centripetal plus Coriolis force vector, $\tau(t) \in \mathbb{R}^n$ is the joint torque vector, and $T_a(t) \in \mathbb{R}^n$ is the unknown disturbance vector which is assumed to be bounded and periodic. And n presents the number of rigid manipulators. The symmetric inertia matrix $D(q(t)) \in \mathbb{R}^{n\times n}$ is assumed to be positive definite and bounded as

$$0 < \lambda_1 I \leq D(q(t)) \leq \lambda_2 I \quad \text{for all } t \in [0, t_f] \tag{3.42}$$

where $\lambda_l, \lambda_2 > 0$ and I is an nxn identity matrix. The matrix $\dot{D}(q(t)) - 2B(q(t), \dot{q}(t))$ is assumed to be skew-symmetric as

$$z^T(\dot{D} - 2B)z = 0 \quad \text{for all } z \in \mathbb{R}^n \text{ and } z \neq 0. \tag{3.43}$$

Denote the desired trajectory $q_d(t) \in \mathbb{R}^n$ is specified as a reference input for the system. The error e_i is defined as

$$e_i = q_d - q_i \tag{3.44}$$

The control law is designed as,

$$\tau_i = E_i + H_i + Y(q(t), \dot{q}(t), \ddot{q}(t))\hat{\Theta} \tag{3.45}$$

where E_i is the feedback input, designed as

$$E_i = \beta L(\dot{e}_i + \alpha e_i) \tag{3.46}$$

where β is a positive constant, L is a symmetric positive definite matrix and α is a positive scale factor.

Define a sliding surface which obtained as

$$s_i = \dot{e}_i + \alpha e_i \tag{3.47}$$

The initial condition, $s_i(0)$ is set to 0, since $e_i(0) = 0$ and $\dot{e}_i(0) = 0$. H_i is the learning law which is proposed as,

$$H_{i+1} = H_i + \beta L s_i \tag{3.48}$$

$Y(q(t), \dot{q}(t), \ddot{q}(t)) \in \mathbb{R}^{n \times l}$ is the regression matrix and $\Theta \in \mathbb{R}^l$ is a suitably chosen parameter vector. $\hat{\Theta}$ is the estimates of the system parameters.The updating law is defined as,

$$\hat{\Theta}_{i+1} = \hat{\Theta}_i + \beta Y s_i \tag{3.49}$$

Theorem 3.1.2 The adaptive learning controller for the uncertain dynamic system uniformly as follows,

(1) $\lim_{i \longrightarrow \infty} V_{i,a}(t) = V_a(t)$

(2) $\lim_{i \longrightarrow \infty} s_i(t) = 0$, for all $t \in [0, t_f]$

where $V_{i,a}(t) = \int_0^t (\tilde{U}_i^T(\tau) L^{-1} \tilde{U}_i(\tau) + \tilde{\Theta}_i^T(\tau) \tilde{\Theta}_i(\tau)) d\tau$ for all $t \in [0, t_f]$ and for all $i \geq 1$.

Proof Since $\|\tilde{\bar{U}}_i\| \geq \|\tilde{U}_i\|$ and $\|\tilde{\bar{\Theta}}_i\| \geq \|\tilde{\Theta}_i\|$, and

$$V_{i+1,a} - V_{i,a} \leq \bar{V}_{i+1,a} - V_{i,a}(t) \tag{3.50}$$

where

$$\bar{V}_{i,a}(t) = \int_0^t (\tilde{\bar{U}}_i^T(\tau) L^{-1} \tilde{\bar{U}}_i(\tau) + \tilde{\bar{\Theta}}_i^T(\tau) \tilde{\bar{\Theta}}_i(\tau)) d\tau \tag{3.51}$$

and

$$\tilde{\bar{\Theta}}_i = \Theta - \hat{\bar{\Theta}}_i \tag{3.52}$$

And define $\Delta\tilde{\Theta}_i$ as $\tilde{\bar{\Theta}}_{i+1} - \tilde{\Theta}_i$, so that

$$\Delta\tilde{\Theta}_i = \tilde{\bar{\Theta}}_{i+1} - \tilde{\Theta}_i = \hat{\Theta}_i - \hat{\bar{\Theta}}_{i+1} = -\beta Y_i s_i \tag{3.53}$$

and

$$\begin{aligned} V_{i+1,a}(t) - V_{i,a}(t) &\leq \bar{V}_{i+1,a} - V_{i,a}(t) \\ &= -2\beta \int_0^t s_i^T (D(q_i)\dot{s}_i + B(q_i, \dot{q}_i) s_i) d\tau \\ &\quad -\beta^2 \int_0^t s_i^T (L + \beta Y_i Y_i^T) s_i d\tau \end{aligned} \tag{3.54}$$

Integrating the first term by part and exploiting the fact that $\dot{D} - 2B$ is skew-symmetric, so that,

$$V_{i+1,a}(t) - V_{i,a}(t) \leq -\beta s_i^T D(q_i) s_i \tag{3.55}$$

where has defined $s_i(0) = 0$. This completes the proof.

3.2 Flexible Manipulator

In recent decades, flexible manipulators are widely used in the industrial field to complete some assembling and transporting tasks. The flexible manipulators can be modeled as Lagrange equation system [65], which is used to describe mechanical and electrical systems. Therefore, solving Lagrange system problems becomes series of hot research topics. At the same time, some space and undersea tasks require the manipulators to be faster rotation speed and less weight with high accuracy. The flexible manipulators get attention because of the lower energy consumption, cost, and higher flexibility than the rigid ones [66]. Because the control input is less than the degrees of freedom of the flexible manipulator, normally, it is treated as under-actuated.

This section aims to develop an adaptive ILC to compensate the effect of dead-zone, which compensates the repetitive disturbance for an under-actuated flexible manipulator. And the flexible manipulator is under-actuated because it is driven

by the torque only at the joint. To compensate the effect of dead-zone, a smooth inverse of dead-zone is used in controller design. The adaptive scheme is used to estimate dead-zone parameters.

3.2.1 Dynamic Modeling

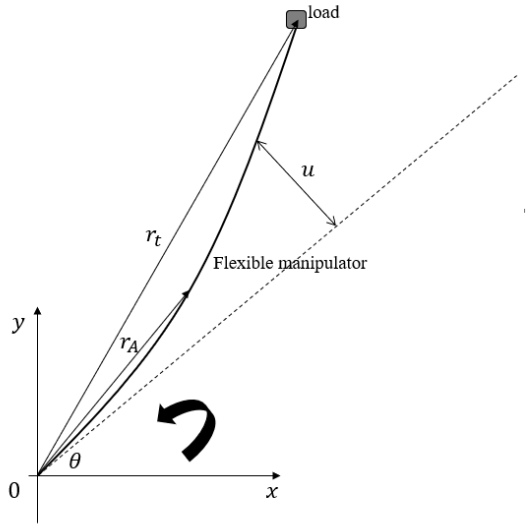


Figure 3.2: Flexible manipulator

As shown in Fig. 3.2. A load is assembled at the tip of a flexible manipulator, and the flexible manipulator can be driven to rotate by connecting to the motor. When the flexible manipulator is rotated, to control the vibration of it directly cannot be achieved. The flexible manipulator can be treated as Euler-Bernoulli beam to simplify the analysis. The kinetic energy and potential energy of this manipulator

system can be written as [65]

$$E_K = \frac{1}{2}\int_0^l \rho \dot{r}_A^T \dot{r}_A dx + \frac{1}{2} m_t \dot{r}_t^T \dot{r}_t \tag{3.56}$$

$$E_p = \frac{1}{2} EI \int_0^l (\frac{\partial^2 u}{\partial x^2})^2 dx \tag{3.57}$$

where r_A and r_t illustrate the position vector of random point on the manipulator and the load, respectively, u is the deformation of the random point on the manipulator, m_t signifies the mass of the load, θ represents the angular position of the manipulator when it rotates, as shown in Fig. 3.2. In the above equations, l, ρ and EI declare the length, density and bending stiffness of the manipulator.

According to the Lagrange equation, the dynamic model of the manipulator shown in Fig. 3.2 can be described as followed

$$M_i(x_i)\ddot{x}_i + C_i(\dot{x}_i, x_i)\dot{x}_i + C_{i,d}(\dot{x}_i, x_i)\dot{x}_i + g_i(x_i) = \tau_i + w \tag{3.58}$$

where i expresses the number of iterations. Since the dimension of the control input is less than the degrees of freedom of the system, the system is treated as under-actuated. Some basic properties of the under-actuated Lagrange system are listed below [65]

Property 1 $M_i = \begin{bmatrix} M_{i,\theta\theta} & M_{i,\theta p} \\ M_{i,p\theta} & M_{i,pp} \end{bmatrix} \in \mathbb{R}^{2\times 2}$ is the inertia matrix, which is positive definite and bounded.

Property 2 $C_i = \begin{bmatrix} C_{i,\theta\theta} & C_{i,\theta p} \\ C_{i,p\theta} & C_{i,pp} \end{bmatrix} \in \mathbb{R}^{2\times 2}$ is the Coriolis and centrifugal matrix and selected by skew-symmetric matrix $\dot{M}_i - 2C_i$.

Property 3 $C_{i,d} = \begin{bmatrix} 0 & 0 \\ 0 & C_{i,d,p} \end{bmatrix} \in \mathbb{R}^{2\times 2}$ is the damping matrix, where $C_{i,d,p} = \eta_1 > 0$.

Property 4 $g_i = \begin{bmatrix} 0 & g_{i,p} \end{bmatrix}^T \in \mathbb{R}^{2\times 1}$ is the stiffness term, $\tau_i = \begin{bmatrix} u_d & 0 \end{bmatrix}^T \in \mathbb{R}^{2\times 1}$ is the control input vector and $w = \begin{bmatrix} w_\theta & w_p \end{bmatrix}^T \in \mathbb{R}^{2\times 1}$ represents the modeling errors and some repetitive disturbances.

Property 5 $x_i = \begin{bmatrix} \theta_i & p_i \end{bmatrix}^T \in \mathbb{R}^{2\times 1}$ is the generalized coordinate vector.

Property 6 ρ, m_t, EI and η_1 are treated as uncertain parameters. Therefore, the dynamic equation is linear with respect to the parameter vector $\Theta_i = \begin{bmatrix} \rho & m_t & EI & \eta_1 \end{bmatrix}^T$ [67, 68].

The dynamic equation can be rewritten as

$$\begin{cases} M_{i,\theta\theta}\ddot{\theta}_i + M_{i,\theta p}\ddot{p}_i + C_{i,\theta p}\dot{p}_i = u_d + w + \theta \\ M_{i,p\theta}\ddot{\theta}_i + M_{i,pp}\ddot{p}_i + C_{i,p\theta}\dot{\theta}_i + (C_{i,pp} + C_{i,d,p}\dot{p}_i) + g_{i,p} = w_p \end{cases} \tag{3.59}$$

According to property, the dynamic equation can be rearranged as

$$\begin{cases} M_{i,\theta\theta}\ddot{\theta}_i + M_{i,\theta p}\ddot{p}_i + C_{i,\theta p}\dot{\theta}_i + C_{i,\theta p}\dot{p}_i = Y_{i,\theta}(x_i, \dot{x}_i, \ddot{\theta}_i, \ddot{p}_i, \dot{\theta}_i, \dot{p}_i)\Theta_i \\ M_{i,p\theta}\ddot{\theta}_i + M_{i,pp}\ddot{p}_i + C_{i,p\theta}\dot{\theta}_i + (C_{i,pp} + C_{i,d,p}\dot{p}_i) + g_{i,p} = Y_{i,p}(x_i, \dot{x}_i, \ddot{\theta}_i, \ddot{p}_i, \dot{\theta}_i, \dot{p}_i)\Theta_i \end{cases} \tag{3.60}$$

where $Y_i = [\ Y_{i,\theta}^T \quad Y_{i,p}^T\ ]^T$ is called the regression matrix.

3.2.2 Dead-zone and Dead-zone inverse

Dead-zone is a phenomenon, which happens during an interval where the output of the control system is zero, while the input of the control system is not zero. In this case, it appears during the flexible manipulator changes the rotation direction, such as from clockwise to counterclockwise. The dead-zone model can be described as [4]

$$u(t) = DZ(v(t)) = \begin{cases} m_r(v(t) - b_r) & v(t) \geq b_r \\ 0 & b_l \leq v(t) \leq b_r \\ m_l(v(t) - b_l) & v(t) \leq b_l \end{cases} \tag{3.61}$$

where b_r, b_l, m_r and m_l are the breakpoints and slope of dead-zone, respectively. $b_r \geq 0$, $b_l \geq 0$, $m_r \geq 0$, $m_l \geq 0$ are unknown constants, $u(t)$ is the control output and $v(t)$ is the control input, as shown in Fig. 3.3.

It should be noted that the magnitudes of b_r and b_l are not required to be equal, and the slope m_r and m_l are not required to be the same.

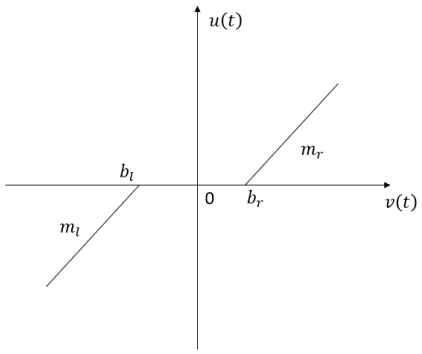


Figure 3.3: Dead-zone model [4]

To remove the effect of dead-zone, the solution in this paper is to place a dead-zone inverse. It is designed to convert the input u_d into control input v. The unknown dead-zone inverse parameters are estimated by the adaptive control scheme and changed follow the number of iterations. Because the dead-zone is unknown, the dead-zone inverse can only be complied with the estimation of dead-zone parameters. The dead-zone inverse is described as

$$v(t) = DI(u_d(t)) = \begin{cases} \frac{u_d(t)+\widehat{m_r b_r}}{\widehat{m_r}}, & \text{if} \quad u_d \geq 0 \\ \frac{u_d(t)+\widehat{m_l b_l}}{\widehat{m_l b_l}}, & \text{if} \quad u_d \geq 0 \end{cases} \tag{3.62}$$

where $\widehat{m_r} \neq 0$, $\widehat{m_r b_r} \neq 0$, $\widehat{m_l} \neq 0$, $\widehat{m_l b_l} \neq 0$ are estimates of the dead-zone parameters m_r, $m_r b_r$, m_l, $m_l b_l$, respectively.

The described dead-zone inverse is a relay-type discontinuity when the parameters for dead-zone inverse is correct, it works and counteracts the effect of dead-zone.

However, when the estimates parameters $\widehat{b_r}$, $\widehat{b_l}$ are different from the true values b_r, b_l of dead-zone, this discontinuity appears and may cause control chattering.

In order to avoid this possible problem, an improved approach which is the smooth inverse of the dead-zone is claimed to compensate for the effect of the dead-zone in controller design. The smooth inverse for the dead-zone is described as

$$v(t) = DI(u_d(t)) = \frac{u_d(t) + \widehat{m_r b_r}}{\widehat{m_r}}\phi_r(u_d) + \frac{u_d(t) + \widehat{m_l b_l}}{\widehat{m_l}}\phi_l(u_d) \tag{3.63}$$

where $\phi_r(u_d)$ and $\phi_l(u_d)$ are smooth continuous functions defined as

$$\begin{cases} \phi_r(u_d) = \frac{e^{u_d/\varepsilon_0}}{e^{u_d/\varepsilon_0}+e^{-u_d/\varepsilon_0}} \\ \phi_l(u_d) = \frac{e^{-u_d/\varepsilon_0}}{e^{u_d/\varepsilon_0}+e^{-u_d/\varepsilon_0}} \end{cases} \tag{3.64}$$

where ε_0 is a free parameter, which can be chosen in any values, but needs to be satisfied greater than 0. The structure of the controller with dead-zone is revealed in Fig. 3.4.

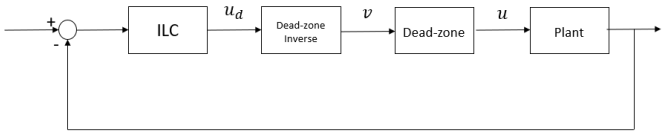


Figure 3.4: Structure of the controller with dead-zone

The error between $u(t)$ and $u_d(t)$ is calculated. The output of the dead-zone $u(t)$

is parameterized as

$$u(t) = -\psi^T \xi \tag{3.65}$$

where

$$\xi = [m_r, m_r b_r, m_l, m_l b_l]^T \tag{3.66}$$

$$\psi = [-\sigma_r(t)v(t), \sigma_r(t), -\sigma_l(t)v(t), \sigma_l(t)]^T \tag{3.67}$$

$$\sigma_r(t) \begin{cases} 1 & \text{if } u(t) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.68}$$

$$\sigma_l(t) \begin{cases} 1 & \text{if } u(t) \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.69}$$

Based on Eq. (3.63), $u_d(t)$ can be written as a function of $v(t)$ as

$$u_d(t) = -\overline{\psi}^T \widehat{\xi} \tag{3.70}$$

where $\hat{\xi}$ denotes estimate of ξ as given below,

$$\hat{\xi} = [\widehat{m_r}, \widehat{m_r b_r}, \widehat{m_l}, \widehat{m_l b_l}]^T \tag{3.71}$$

$$\overline{\psi} = [-\phi_r(t)v(t), \phi_r(t), -\phi_l(t)v(t), \phi_l(t)]^T \tag{3.72}$$

Additionally, $\hat{\xi}$ is obtained by the following equation

$$\dot{\hat{\xi}} = -\Gamma \overline{\psi}^T s \tag{3.73}$$

where Γ is a positive constant.

3.2.3 Control System Design

The control objective is to drive an under-actuated flexible manipulator with uncertain parameters, repeatable disturbance and dead-zone to track a desired trajectory. Some assumptions are imposed [65].

Assumption 1. The desired angular trajectory is defined by $\overline{\theta}_d(t)$ and gets 2 requirements.

(1) $\overline{\theta}_d(t)$ is a bounded continuously differentiable function of time t.

(2) The speed and acceleration are bounded, which means $\dot{\overline{\theta}}_d(t)$ and $\ddot{\overline{\theta}}_d(t)$ are bounded.

Assumption 2. The initial values of the angular position $\overline{\theta}_d(t)$ and angular velocity $\dot{\overline{\theta}}_d(t)$ of the flexible manipulator are the same as the desired initial state at the beginning of each iteration, which means $\theta_i(0) = \overline{\theta}_d(0)$ and $\dot{\theta}_i(0) = \dot{\overline{\theta}}_d(0)$. Furthermore, the unknown variable p_i has the same initial value during each iteration, which is named as the Iterative Learning Control [69].

Assumption 3. The modeling errors and repetitive disturbance are bounded.

Determine $e_i = \theta_i - \overline{\theta}_d$ presents the tracking error. The initial velocities of θ_i and

p_i are $\dot{\overline{\theta}}_{(i,r)}$ and $\dot{p}_{(i,r)}$, which are obtained from

$$\dot{\theta}_{i,r} = \lambda e_i \tag{3.74}$$

$$Y_{i,p}(x_i, \dot{x}_i, \ddot{\theta}_i, \ddot{p}_i, \dot{\theta}_i, \dot{p}_i)\hat{\Theta}_i = \alpha \tanh(\dot{p}_i - \dot{p}_{i,r}) + \hat{w}_{i,p} \tag{3.75}$$

where λ and α are positive constant, $\hat{\Theta}_i$ and $\hat{w}_{i,p}$ are the estimates of Θ_i and $w_{i,p}$. Here defined two auxiliary variables, which are called sliding surfaces. The starting value $\dot{p}_{(i,r)}$ is set as $\dot{p}_i(0)$. Based on the Assumption 2 and the meaning of $\dot{p}_i$, initially, s_i and r_i are zero.Two variables are defined as

$$s_i = \dot{\theta}_i - \dot{\theta}_{i,r} \tag{3.76}$$

$$r_i = \dot{p}_i - \dot{p}_{i,r} \tag{3.77}$$

so that it can be obtained as

$$\begin{cases} M_{i,\theta\theta}\dot{s}_i + M_{i,\theta p}\dot{r}_i + C_{i,\theta p}s_i + C_{i,\theta p}r_k = \tau_{i,\theta} + w_\theta - Y_{i,\theta}(x_i, \dot{x}_i, \ddot{\theta}_{i,r}, \ddot{p}_{i,r}, \dot{\theta}_{i,r}, \dot{p}_{i,r})\Theta_i \\ M_{i,p\theta}\dot{s}_i + M_{i,pp}\dot{r}_i + C_{i,p\theta}s_i + (C_{i,pp} + C_{i,d,p}r_i) = w_p - Y_{i,p}(x_i, \dot{x}_i, \ddot{\theta}_{i,r}, \ddot{p}_{i,r}, \dot{\theta}_{i,r}, \dot{p}_{i,r})\Theta_i \end{cases} \tag{3.78}$$

The tracking controller is designed as follows

$$\tau_{i,\theta} = -\mu \tanh(\beta s_i) + Y_{i,\theta}(x_i, \dot{x}_i, \ddot{\theta}_{i,r}, \ddot{p}_{i,r}, \dot{\theta}_{i,r}, \dot{p}_{i,r})\hat{\Theta}_i - \hat{w}_{i,\theta} \tag{3.79}$$

where μ and β are positive constant, $\hat{w}_{i,\theta}$ is the estimates value of $w_(i, \theta)$. $\hat{w}_{(i,p)}$ and $\hat{w}_{(i,\theta)}$ are obtained from the following learning lows and the initial value of $\hat{w}_{(i,p)}$

and $\hat{w}_{(i,\theta)}$ are both equal to zero

$$\hat{w}_{(i,\theta)} = \hat{w}_{(i-1,\theta)} + \xi s_i \tag{3.80}$$

$$\hat{w}_{(i,p)} = \hat{w}_{(i-1,p)} + \xi r_i \tag{3.81}$$

where ξ is a positive constant learning gain.

Denote $\Theta_{i,u} = -(Y_{i,\theta}^T s_i + Y_{i,p}^T r_i)$. The $j-th$ entry of $\hat{\Theta}_i$, i.e., $\hat{\Theta}_{i,j}$, is updated by the following law,

$$\dot{\hat{\Theta}}_{i,j} = \begin{cases} \gamma \Theta_{i,u,j} & if \hat{\Theta}_{i,j} \geq \varepsilon \\ \gamma \sqrt{\Theta_{i,u,j}^2 + \varepsilon_1} & if \hat{\Theta}_{i,j} < \varepsilon \end{cases} \tag{3.82}$$

where $\Theta_{i,u,j}$ is the $j-th$ element of $\Theta_{i,u}$, both ε and ε_1 are small positive numbers and γ is a positive constant gain. ε is chosen very small so that it is less then every entry of $\Theta_{i,j}$. The initial condition and iteration initial guess of $\hat{\Theta}_i$ are

$$\hat{\Theta}_i(0) = \hat{\Theta}_{i-1}(T), \hat{\Theta}_1(0) = \delta \tag{3.83}$$

where δ is a positive number and T is a finite large number.

3.3 Simulation Results

This section mainly presents some simulation results for trajectory tracking of a rigid manipulator, using PD-type and adaptive-type ILCs. All the simulation results are obtained solely using MATLAB & Simulink programming.

3.3.1 PD-Type ILC

Throughout the simulation, the given desired trajectory is set as

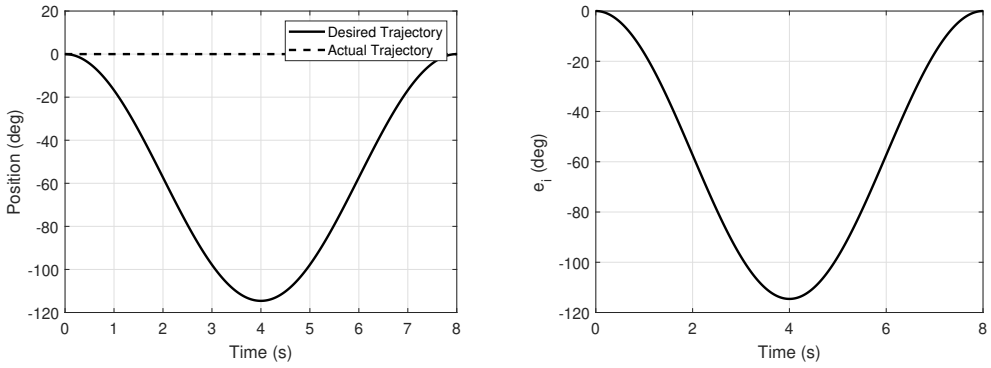
$$\sin(\frac{\pi}{4}t + \frac{\pi}{2}) - 1 \tag{3.84}$$

where $t = 8$ s is the iteration time. And the learning control gain Γ and R are chosen as

$$\Gamma = 0.8 \tag{3.85}$$

$$R = -0.12 \tag{3.86}$$

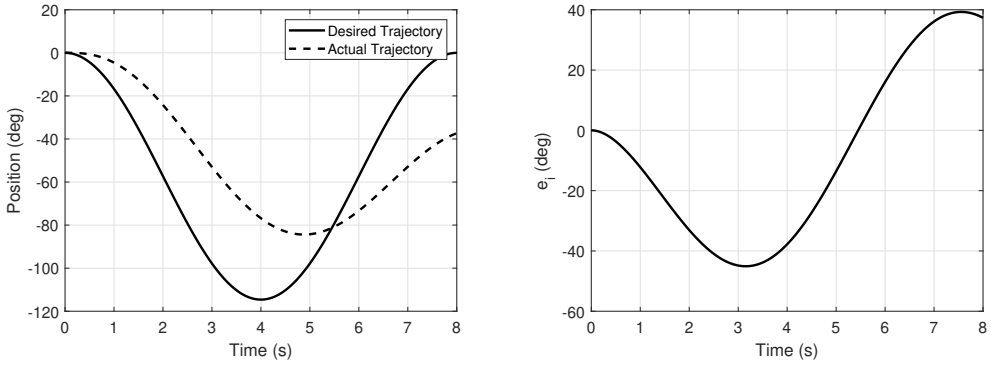
Because $u_0(t)$, $e_0(t)$ and $\dot{e}_0(t)$ are set as 0, there is no response at the first iteration, and the results are shown as Fig. 3.5. After the first iteration, $u_1(t)$, $e_1(t)$ and $\dot{e}_1(t)$



(a) Tracking performance (b) Tracking error

Figure 3.5: PD-type ILC simulation results : trajectory tracking at the 1^{st} iteration

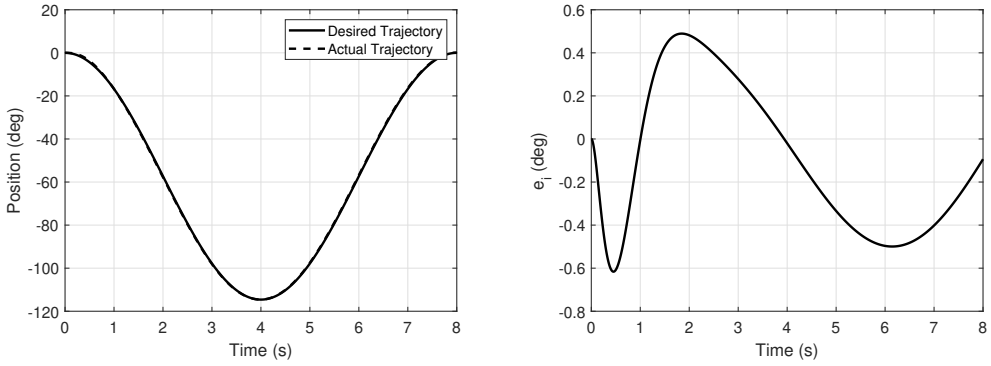
are updated, then the results in the second iteration is as followed Fig. 3.6.



(a) Tracking performance (b) Tracking error

Figure 3.6: PD-type ILC simulation results : trajectory tracking at the 2^{nd} iteration

After taking 9 iterations, the desired trajectory gradually converges to the desired trajectory, and the tracking error is less than ± 0.6 deg of the desired trajectory, as shown in Fig. 3.7. Therefore, according to the shown simulation results, the rigid manipulator tracks the desired trajectory after 10 iterations by using PD-type ILC algorithm.



(a) Tracking performance (b) Tracking error

Figure 3.7: PD-type ILC simulation results : trajectory tracking at the 10^{th} iteration

3.3.2 Adaptive-Type ILC

Similarly, throughout the simulation, the given desired trajectory is set as

$$\sin(\frac{\pi}{4}t + \frac{\pi}{2}) - 1 \tag{3.87}$$

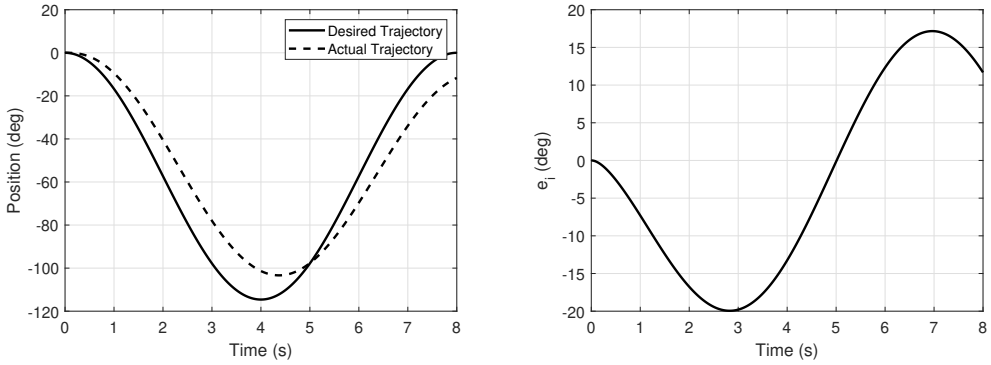
where $t = 8$ s is the iteration time. And the learning control gain β, L and α are chosen as

$$\beta = 1 \tag{3.88}$$

$$L = 2 \tag{3.89}$$

$$\alpha = \frac{1}{3} \tag{3.90}$$

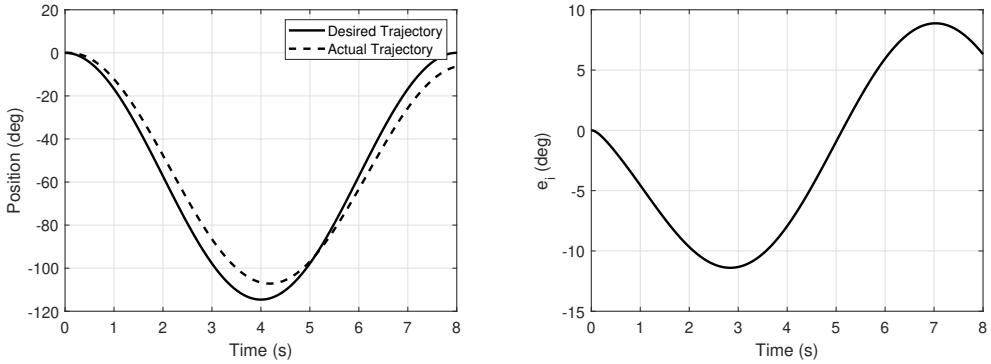
The initial learning control input $H_1(t)$, sliding surface $s_1(t)$, $Y_1(t)$ and estimated parameters $\widehat{\Theta}_1(t)$ are all chosen as 0. As seen in Fig. 3.8, the trajectory tracking at the first iteration including tracking response (Fig. 3.8(a)) and tracking error (Fig. 3.8(b)). The difference between PD-Type and Adaptive-Type is there is control response at the first iteration, due to the feedback term in adaptive-Type ILC controller.



(a) Tracking performance (b) Tracking error

Figure 3.8: Adaptive-type ILC simulation results : trajectory tracking at the 1^{st} iteration

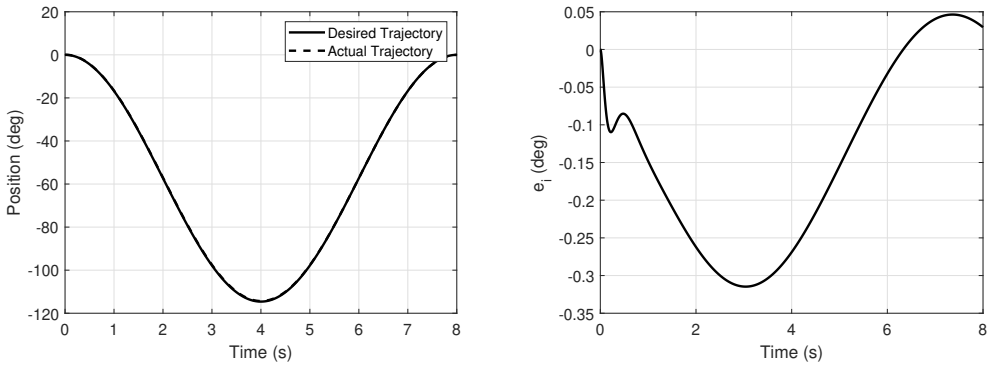
Then, from Fig. 3.9, the actual trajectory gives the trend to converge to the desired trajectory, the tracking error decreased as well.



(a) Tracking performance (b) Tracking error

Figure 3.9: Adaptive-type ILC simulation results : trajectory tracking at the 2^{nd} iteration

At the 9^{th} iteration, the tracking error has decreased to between -0.35 deg and 0.05 deg. According to the given simulation results, adaptive-type ILC controller drive the rigid manipulator tracks the desired trajectory using 9 iterations.



(a) Tracking performance (b) Tracking error

Figure 3.10: Adaptive-type ILC simulation results : trajectory tracking at the 9^{th} iteration

3.4 Experimental Results

3.4.1 Rigid Manipulator

Rigid manipulator experimental results are presented in this section. As shown in Fig. 3.11, the experimental system consists of a Q2-USB acquisition device, an amplifier device, a computer, and a Quanser SRV02 rotary servo plant. A rigid link is mounted on the top of Quanser SRV02 rotary servo plant [70]. The angular position is detected by encoder sensor.

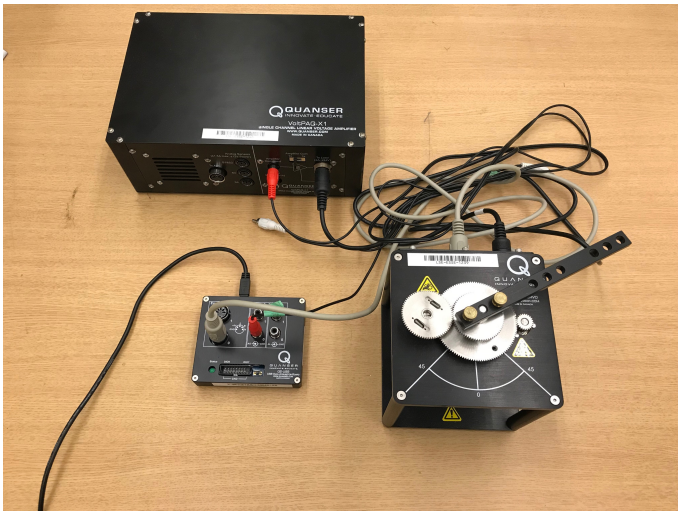
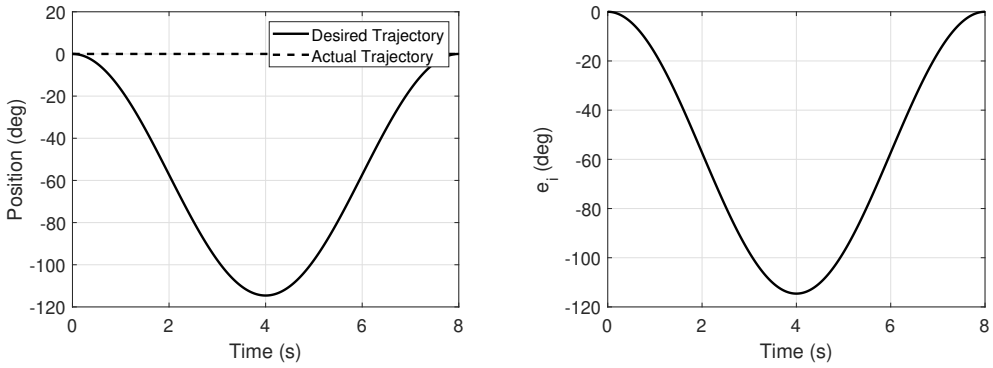


Figure 3.11: Rigid manipulator experimental system

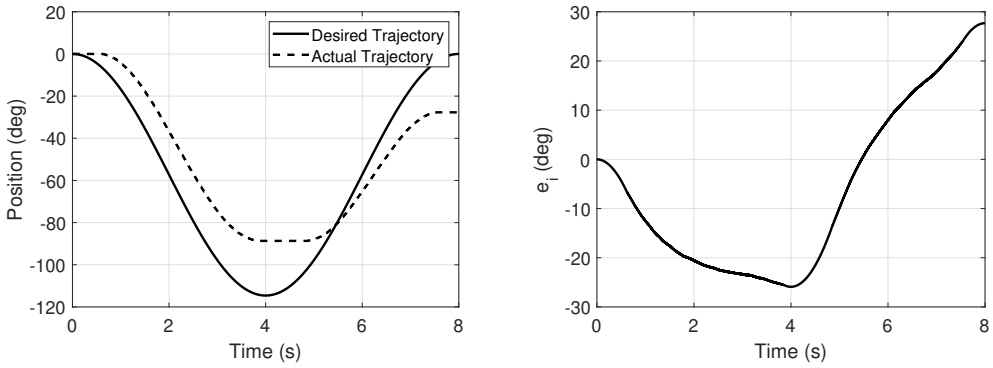
3.4.1.1 PD-Type ILC

In each iteration process, the experiment time, T, is set as 8 seconds. Fig. 3.12 shows the desired trajectory (solid line) and actual trajectory (dash line) at the 1^{st} iterations. It is clear to see that the gap between the desired trajectory and actual trajectory at the 1^{st} iteration. Then, at the 2^{nd} iteration, e_i and $\dot{e}_1$ are updated, in this case, there is a response that the rigid manipulator start to track the desired trajectory, as shown in Fig. 3.13. Finally, actual trajectory converges to the desired trajectory at the 14^{th} iteration. And Fig. 3.14 presents the tracking error at the 14^{th} iteration, which obviously tells the tracking performances are improved as iteration time increased.



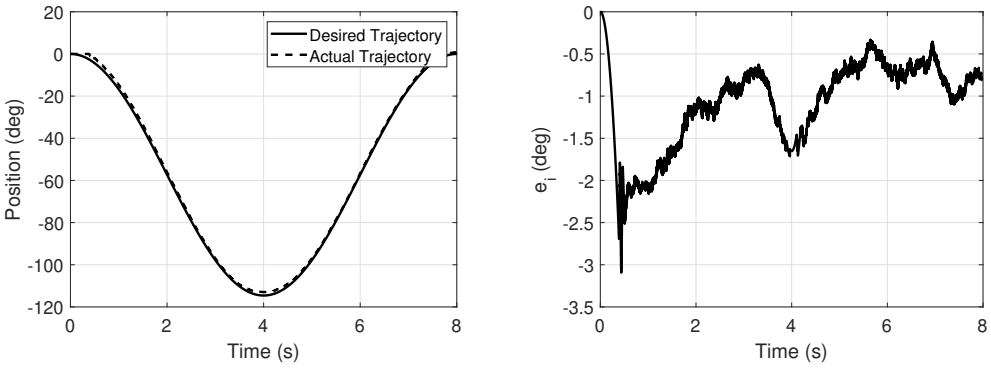
(a) Tracking performance (b) Tracking error

Figure 3.12: PD-type ILC experimental results : trajectory tracking at the 1^{st} iteration



(a) Tracking performance (b) Tracking error

Figure 3.13: PD-type ILC experimental results : trajectory tracking at the 2^{nd} iteration

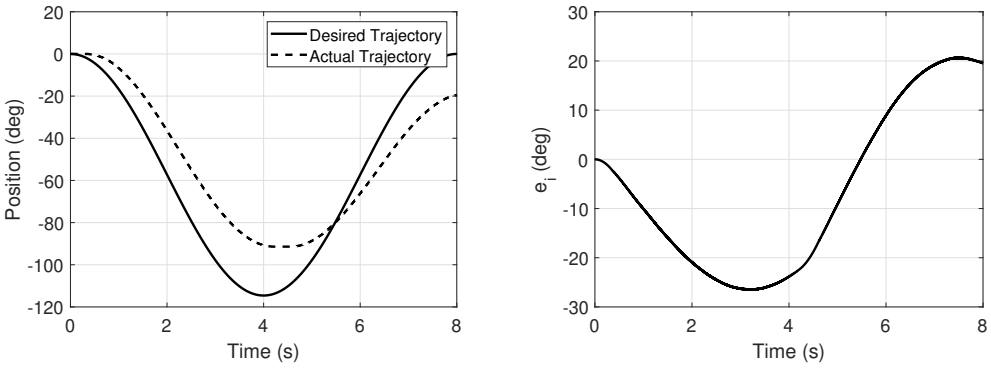


(a) Tracking performance (b) Tracking error

Figure 3.14: PD-type ILC experimental results : trajectory tracking at the 14^{th} iteration

3.4.1.2 Adaptive-Type ILC

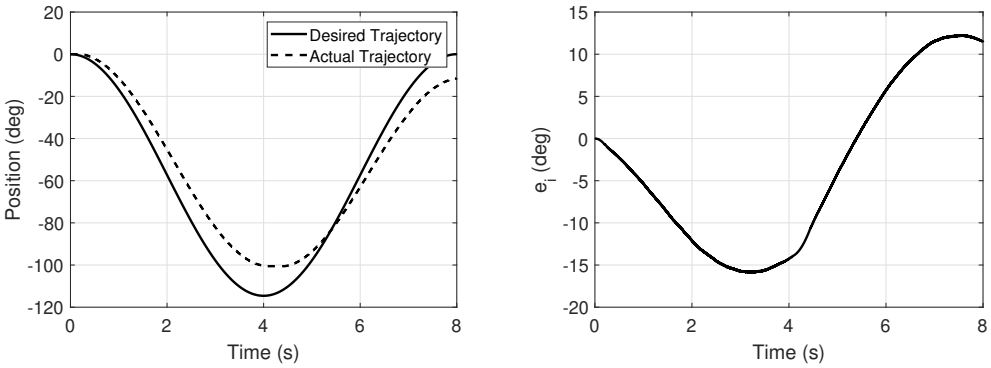
In each iteration process, the experiment time, T, is set as 8 seconds. Fig. 3.15 shows the trajectory and tracking error at the 1^{st} iterations. And the given trajectory is the same as simulation.



(a) Tracking performance (b) Tracking error

Figure 3.15: Adaptive-type ILC experimental results : trajectory tracking at the 1^{st} iteration

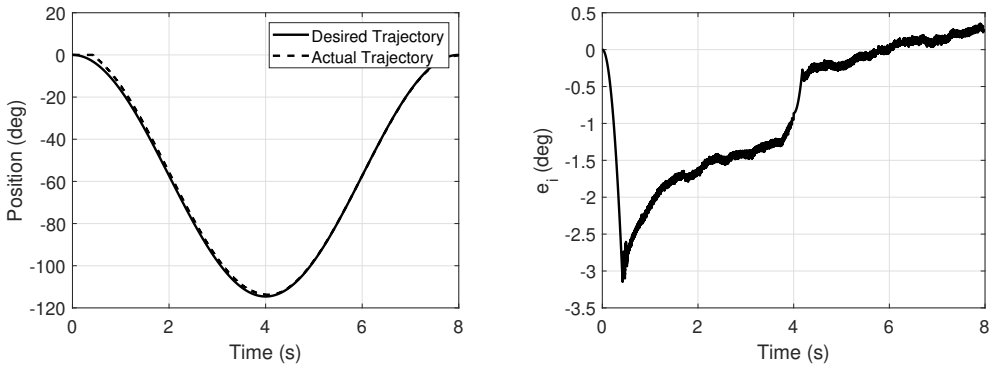
Then the 2^{nd} iteration is shown as Fig. 3.16.



(a) Tracking performance (b) Tracking error

Figure 3.16: Adaptive-type ILC experimental results : trajectory tracking at the 2^{nd} iteration

Finally, at the 8^{th} iteration, the tracking error is within 0.5 deg, as shown in Fig. 3.17.



(a) Tracking performance (b) Tracking error

Figure 3.17: Adaptive-type ILC experimental results : trajectory tracking at the 8^{th} iteration

3.4.2 Flexible Manipulator

Flexible manipulator experimental results are presented in this section. As shown in Fig. 3.18, the experimental system has four parts, including a Q2-USB acquisition device, an amplifier device, a computer, and a Quanser SRV02 rotary servo plant. A flexible link with a tip mass is assembled on the top of Quanser SRV02 rotary servo plant. The angular position is detected by encoder sensor and deformation is detected by a strain gage connected to the tachometer sensor. The parameters of the flexible manipulator, control system and dead-zone are shown in Table 3.1

[71]. In each iteration process, the experiment time, T, is set as 40 seconds.

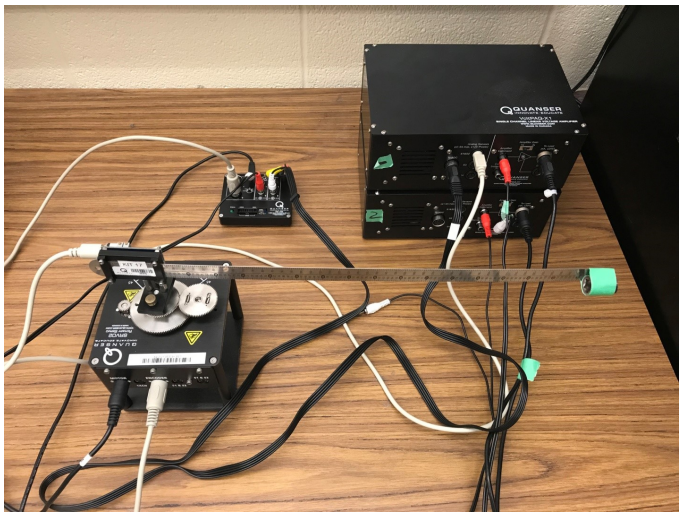


Figure 3.18: Flexible manipulator experimental system

The designed trajectory is set as $\bar{\theta}_d = \sin(\frac{\pi}{10}T + \frac{\pi}{2}) - 1$. In the actuated channel, manually subjoin a $0.1\sin(1.5T)$ disturbance to demonstrate the performance of the designed controller. Fig. 3.19 shows the s at the 1st and 10th iterations. It is clear to see that s gradually reduce to almost zero as iteration time increasing. The trajectories of the flexible manipulator at 1st and 10th iterations are illustrated in Fig. 3.20. And Fig. 3.21 presents the tracking error at the 1st and 10th iterations, which obviously tell the tracking performances are promoted from time by time.

Table 3.1: Parameters of system, controller and deadzone

Parameters	Value	Unit
Linear density of flexible link ρ	0.1354	kg/m
Length of flexible link l	0.435	m
Bending stiffness of flexible link EI	0.181	N·m^2
Mass of tip payload m_t	0.0324	kg
η_1	0.01	-
λ	5	-
α	0.1	-
ϵ	10^{-3}	-
ϵ_1	10^{-5}	-
β	2	-
γ	0.5	-
δ	0.1	-
φ	0.5	-
μ	0.2	-
ε_0	0.01	-
$\widehat{m_r}$	1	-
$\widehat{m_r b_r}$	0.01	-
$\widehat{m_l}$	1	-
$\widehat{m_l b_l}$	0.01	-

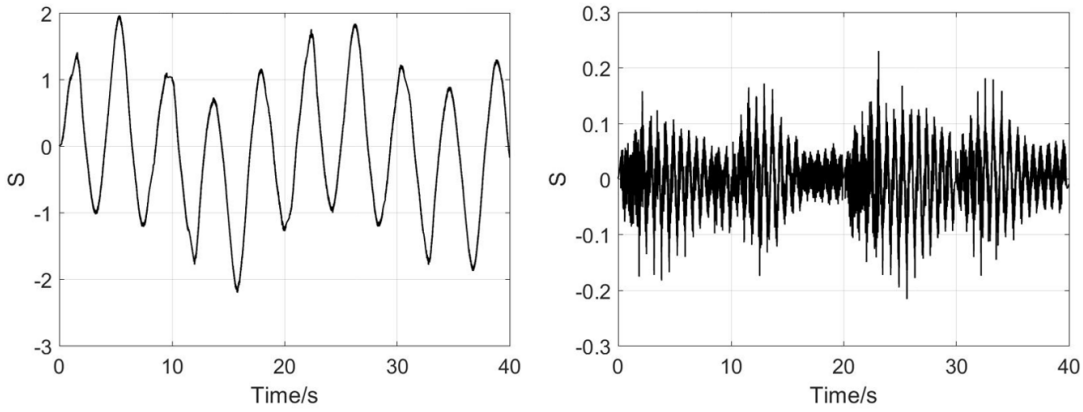


Figure 3.19: The sliding variable s at the 1^{st} and 10^{th} iteration

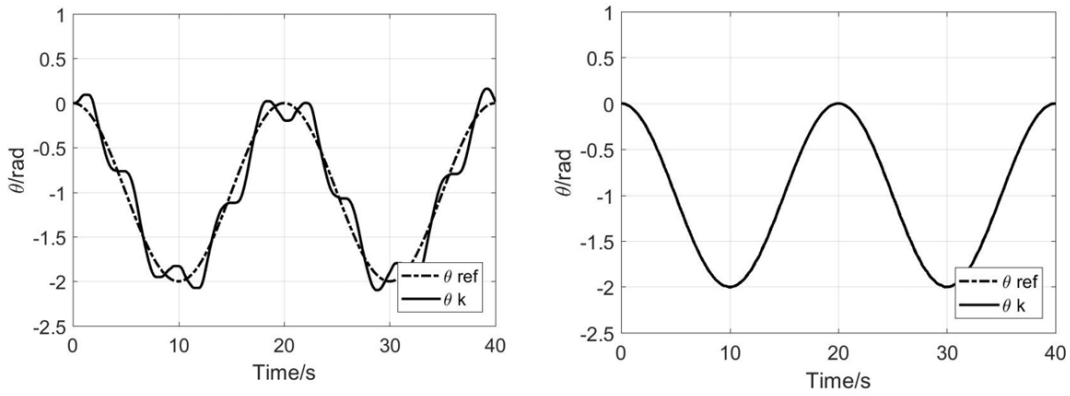


Figure 3.20: The trajectory tracking of the flexible manipulator at the 1^{st} and the 10^{th} iteration with dead-zone inverse

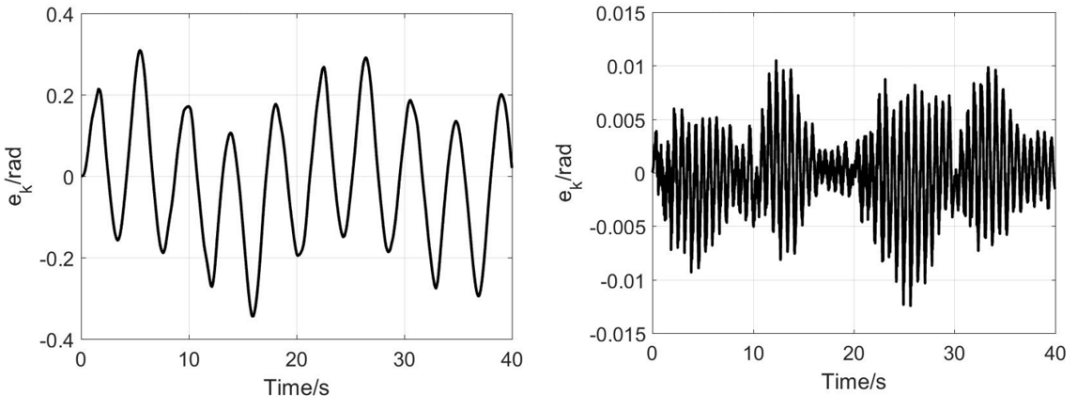


Figure 3.21: The tracking error at the 1^{st} and the 10^{th} iteration with dead-zone inverse

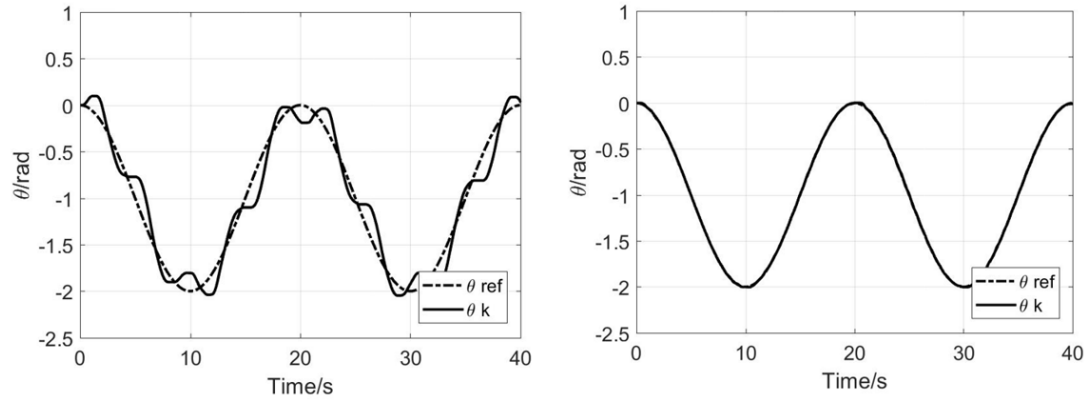


Figure 3.22: The trajectory tracking of the flexible manipulator at the 1^{st} and the 10^{th} iteration without dead-zone inverse

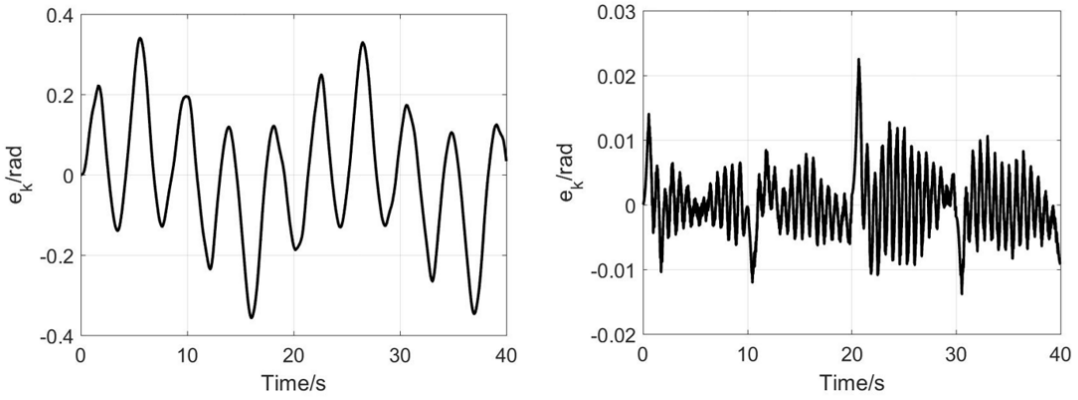


Figure 3.23: The tracking error at the 1^{st} and the 10^{th} iteration without dead-zone inverse

All above results contain the smooth inverse of dead-zone. Moreover, Fig. 3.22 clarifies the trajectories of the flexible manipulator at 1^{st} and 10^{th} iterations, which do not have dead-zone inverse. Additionally, Fig. 3.23 indicates the tracking error at the 1^{st} and 10^{th} iterations. By comparing with Fig. 3.20, Fig. 3.21, Fig. 3.22 and Fig. 3.23, it is should be noted that placing the smooth inverse of dead-zone in controller design can successfully compensate the effect of dead-zone.

3.5 Conclusions

Table 3.2: Comparison of PD-type ILC and adaptive-type ILC

	PD-Type ILC	Adaptive-Type ILC
Simulation tracing error (deg)	$\mid e_i \mid \leq 0.6$	$\mid e_i \mid < 0.35$
Simulation converge speed	10 iterations	9 iterations
Experimental tracking error (deg)	$\mid e_i \mid < 1.5$	$\mid e_i \mid < 1$
Experimental converge speed	14 iterations	8 iterations
First iteration	no response	tracking

In this chapter, dynamic model of both rigid and flexible manipulator are presented, and based on these two dynamic models, two types of ILC algorithm are designed to ensure the manipulator tracks the desired trajectories, and compensates repetitive disturbance. As shown in simulation and experimental results, the proposed PD-type ILC and adaptive-type ILC meet the requirements, which is the tracking error is less than 1% of the maximum value of the desired trajectory. By compared with these two controllers, PD-type has the lower convergence speed, and at the first iteration, there is no response which causes some misunderstanding in applications, such as the operator may think the instrument is broken and waste 1 time to manipulate the materials. For the Adaptive-type ILC, tracking error at the 1st iteration is acceptable, and it took 8 iterations to converge to the required speci-

fications. And from Table 3.2, it can be seen the tracking error for adaptive-type ILC is less than that for PD-type ILC in both simulation and experimental results.

The problem of a flexible manipulator with uncertain parameters, dead-zone, and unknown disturbance is addressed using the designed iterative learning controller with placing a smooth inverse of the dead-zone. An adaptive sliding-mode controller keeps the system on the designed sliding variables during the experimental time interval as iteration time goes up. The smooth inverse of the dead-zone is designed to compensate the effect of the dead-zone. Experimental results show that the proposed controller can drive the flexible manipulator tracking the designed trajectory and compensate the effect of the dead-zone.

4 Application of ILC to UAVs

In this chapter, detailed discussions of a quadrotor UAV are presented. The organization of this chapter is as follows: the dynamic model of the UAV, the basic concepts and the assumptions used in this chapter are introduced in Section 4.1. The controller design based on the attitude and the position control is explained in 4.2. The simulation of the quadrotor is discussed in Section 4.3. The experimental results using gimbal and QDrone are shown in section 4.4 and finally concluding remarks of this chapter are included in Section 4.5.

4.1 Dynamic Modeling

The UAV configuration [72] used in this model is illustrated in Fig. 4.1. As seen in the figure, based on the right hand rule, the three rotations Roll (ϕ), Pitch (θ) and Yaw (ψ) present the rotation about X-axis, Y-axis and Z-axis, respectively. Propellers 1 and 2 rotate in clockwise, meanwhile generate the torques 1 and 2.

Similarly, propellers 3 and 4 rotate in counter-clockwise and generate torques 3 and 4 , which are in the opposite direction of torques 1 and 2 to keep the stability of the UAV.

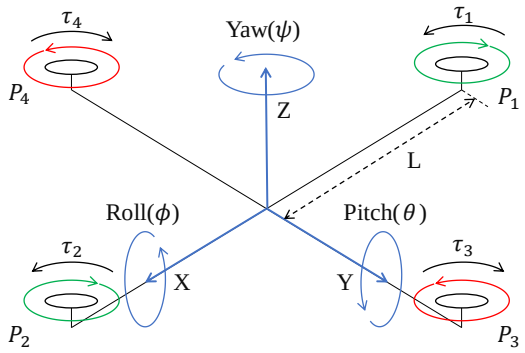


Figure 4.1: UAV schematic

The dynamic model fo a UAV with respect to the Earth-fixed coordinate system can be expressed as [73]

$$\begin{cases} \ddot{x} = (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\frac{U_1}{m} - \frac{k_1}{m}\dot{x} \\ \ddot{y} = (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\frac{U_1}{m} - \frac{k_2}{m}\dot{y} \\ \ddot{z} = (\cos\phi\cos\theta)\frac{U_1}{m} - g - \frac{k_3}{m}\dot{z} \\ \ddot{\phi} = \frac{U_2}{I_x} - L\frac{k_4}{I_x}\dot{\phi} \\ \ddot{\theta} = \frac{U_3}{I_y} - L\frac{k_5}{I_y}\dot{\theta} \\ \ddot{\psi} = \frac{U_4}{I_z} - L\frac{k_6}{I_z}\dot{\psi} \end{cases} \tag{4.1}$$

with

$$\begin{cases} U_1 = F_1 + F_2 + F_3 + F_4 \\ U_2 = L(F_3 - F_4) \\ U_3 = L(F_1 - F_2) \\ U_4 = K_y(F_1 + F_2 - F_3 - F_4) \end{cases} \tag{4.2}$$

where, x, y, and z represent the coordinates in the inertial frame, ϕ, θ, and ψ illustrate roll, pitch, and yaw angles, k_i represents the drag coefficients, m, g, F_i, and K_y are the mass of the UAV, acceleration of gravity, thrust generated be the ith motor and thrust-to-moment scalling factor, respectively, and U_1, U_2, U_3, U_4, and L illustrate the total lift force, the moments about x, y, and z axes, and the distance between the center of the UAV and the rotor.

To simplify the nonlinear model shown above, the linear model of a UAV can be expressed with an assumption:

Assumption It is assumed that the UAV operates in a hovering condition ($U_1 \approx mg$). The pitch and roll motions are small such that $\sin\theta \approx \theta$ and $\sin\phi \approx \phi$. There is no yaw motion ($\psi = 0$) during the whole flight. Furthermore, the UAV moves extremely slow, so the drag force can be neglected [73].

Based on the assumption, the simplified dynamic model can be obtained in the

following equation:

$$\begin{cases} \ddot{x} = \theta g \\ \ddot{y} = -\phi g \\ \ddot{z} = \frac{U_1}{m} - g \\ \ddot{\phi} = \frac{U_2}{I_x} \\ \ddot{\theta} = \frac{U_3}{I_y} \\ \ddot{\psi} = \frac{U_4}{I_z} \end{cases} \tag{4.3}$$

The mathematical model of UAV with unknown disturbance could be expressed as

$$\boldsymbol{D\ddot{q} + U_a = U} \tag{4.4}$$

where $\boldsymbol{U_a} = [U_{a1},\ U_{a2},\ U_{a3},\ U_{a4}]^T$ is the unknown disturbance vector.Denote $\boldsymbol{q_i} = [x_i,\ y_i,\ z_i,\ \phi_i,\ \theta_i,\ \psi_i]^T$ as the actual output, and $\boldsymbol{q_d} = [x_d,\ y_d,\ z_d,\ \phi_d,\ \theta_d,\ \psi_d]^T$ is specified as a reference input for system, e_i is the tracking error, which is defined as $\boldsymbol{e_i} = \boldsymbol{q_i} - \boldsymbol{q_d}$, and $\dot{\boldsymbol{e}}_{\boldsymbol{i}} = \dot{\boldsymbol{q}}_{\boldsymbol{i}} - \dot{\boldsymbol{q}}_{\boldsymbol{d}}$.

Z-direction Translation

Taking off and raising the height for a quadrotor can be done by increasing the rotors' speeds equally, and vice versa for landing and reducing the height. The change in the torque in each pair is equal and thus be canceled out. Fig. 4.2 illustrates the concept of the vertical motion.

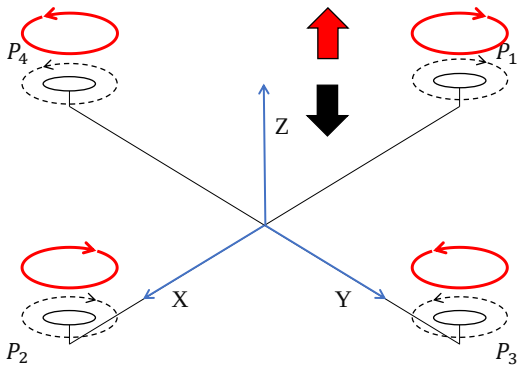


Figure 4.2: Z-direction translation schematic

X-direction Translation and Pitch Rotation

Since the quadrotor dynamic is highly coupled, the motion in one direction depends on a rotation with a certain angle, which is the fundamental of how the quadrotor moves in each direction. Therefore, starting from hovering, increasing the speed of rotor 1 and decreasing the speed of rotor 2, while maintaining speeds of rotor 3 and 4 result in rotation in the pitch(θ) angle and motion in the X-direction and vice versa. Note that at this point the UAV tilts with a small angle and the thrust is approximately equal to weight, thus no motion in the Z-direction. This is illustrated in Fig. 4.3.

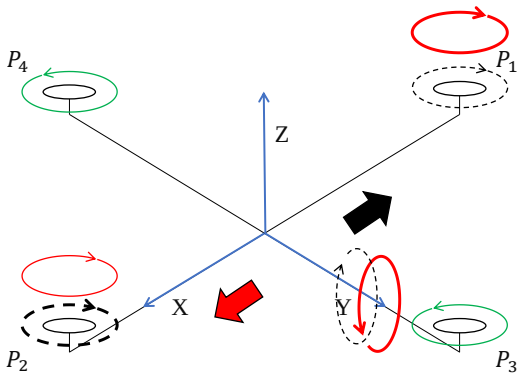


Figure 4.3: X-direction translation and pitch rotation schematic

Y-direction Translation and Roll Rotation

Similar to the pitch motion, the roll motion is coupled with the Y-direction motion. In this case, increasing the rotors' speed of rotor 3 and decreasing it of rotor 4, while maintaining the speeds of rotors 1 and 2 result in rotation in the roll(ϕ) angle and motion in Y direction and vice versa. This is described in Fig. 4.4.

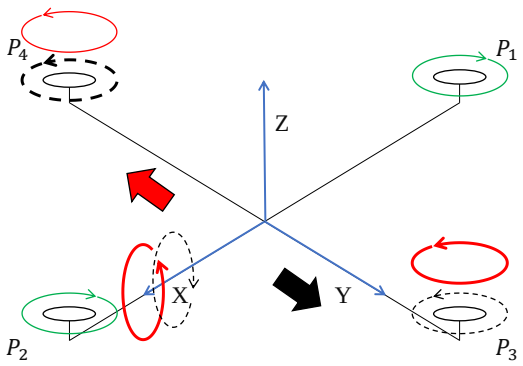


Figure 4.4: Y-direction translation and roll rotation schematic

Yaw-direction Rotation

The yaw motion is not coupled. Instead of canceling out the torques, the thrust is balanced to maintain the altitude in this case. To perform a pure yaw motion, starting from hovering, equally increase the speed in a pair of rotors with the same direction of propeller rotation, and decrease the other pair. Increasing the speed of rotors 1 and 2, while decreasing the speed of rotor 3 and 4 result in rotation in yaw(ψ) direction and vice versa. See Fig. 4.5 for more details.

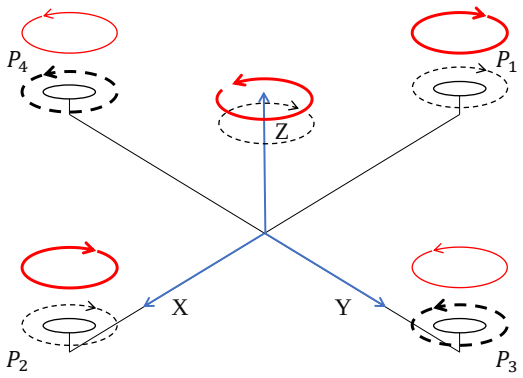


Figure 4.5: Yaw-direction rotation schematic

4.2 Control System Design

4.2.1 Controller Design Based on the Attitude Control

The attitude control considers the attitude of UAV, which are ψ, θ and ψ, Z direction still be considered in this part. Therefore, attitude equations are used as the dynamic model.

4.2.1.1 Learning control without parameter adaptation

$\boldsymbol{U} = [U_1,\ U_2,\ U_3,\ U_4]^T$ is the control input, which can be expressed as follows

$$\boldsymbol{U} = \boldsymbol{E_i} + \boldsymbol{C_i} + \boldsymbol{H_i}, \tag{4.5}$$

where, $\boldsymbol{E_i} = -\beta L\boldsymbol{S}$ is the feedback control input, $\boldsymbol{C_i} = \boldsymbol{D\dot{e}_i}$ is the nonlinear compensation term. The four sliding surfaces [74] are defined as

$$\begin{cases} s_1 = (z_i - z_d) + (\dot{z}_i - \dot{z}_d) \\ s_2 = (y_i - y_d) + (\dot{y}_i - \dot{y}_d) \\ s_3 = (x_i - x_d) + (\dot{x}_i - \dot{x}_d) \\ s_4 = (\psi_i - \psi_d) + (\dot{\psi}_i - \dot{\psi}_d) \end{cases} \tag{4.6}$$

Based on Eq. 4.6, can define $\boldsymbol{S} = [s_1,\ s_2,\ s_3,\ s_4]^T$ and $\boldsymbol{\dot{S}} = [\dot{s}_1,\ \dot{s}_2,\ \dot{s}_3,\ \dot{s}_4]^T$, $\boldsymbol{\dot{S}}$ expressed as

$$\begin{cases} \dot{s}_1 = (\dot{z}_i - \dot{z}_d) + (\ddot{z}_i - \ddot{z}_d) \\ \dot{s}_2 = (\dot{y}_i - \dot{y}_d) + (\ddot{y}_i - \ddot{y}_d) \\ \dot{s}_3 = (\dot{x}_i - \dot{x}_d) + (\ddot{x}_i - \ddot{x}_d) \\ \dot{s}_4 = (\dot{\psi}_i - \dot{\psi}_d) + (\ddot{\psi}_i - \ddot{\psi}_d) \end{cases} \tag{4.7}$$

Substitute Eq. (4.5) into Eq. (4.4), could have

$$\boldsymbol{D\ddot{q}_i} + \boldsymbol{U_a} = \boldsymbol{E_i} + \boldsymbol{C_i} + \boldsymbol{H_i} \tag{4.8}$$

$$\boldsymbol{D}(\boldsymbol{\ddot{e}_i} + \boldsymbol{\ddot{q}_d}) + \boldsymbol{U_a} = -\beta L\boldsymbol{S} + \boldsymbol{D\dot{e}_i} + \boldsymbol{H_i} \tag{4.9}$$

therefore,

$$\boldsymbol{D\dot{S}} + \beta L\boldsymbol{S} = -(\boldsymbol{D\ddot{q}_d} + \boldsymbol{U_a} - \boldsymbol{H_i}) = -(\boldsymbol{U_d^*} + \boldsymbol{U_a} - \boldsymbol{H_i}) = -(\boldsymbol{U_d} - \boldsymbol{H_i}) = \boldsymbol{\tilde{U}_i} \tag{4.10}$$

To generate the learning algorithm, define a Lyapunov function candidate $W(\boldsymbol{S_i}) = \frac{1}{2}\boldsymbol{S_i}^T\boldsymbol{D}\boldsymbol{S_i}$, then, the derivative of $W(\boldsymbol{S_i})$ along the error trajectory is

$$\dot{W}(\boldsymbol{S_i}) = \boldsymbol{S_i}^T\boldsymbol{D}\dot{\boldsymbol{S}}_i + \frac{1}{2}\boldsymbol{S_i}^T\dot{\boldsymbol{D}}\boldsymbol{S_i} = -\beta\boldsymbol{S_i}^T L\boldsymbol{S_i} + \boldsymbol{S_i}^T\tilde{\boldsymbol{U}}_i \tag{4.11}$$

Integrating both sides of Eq. (4.11), results in

$$W(\boldsymbol{S_i}(t)) - W(\boldsymbol{S_i}(0)) = -\int_0^t \beta\boldsymbol{S_i}^T L\boldsymbol{S_i}d\tau + \int_0^t \boldsymbol{S_i}^T\tilde{\boldsymbol{U}}_i d\tau \tag{4.12}$$

which indicates that the error dynamics is strictly passive with respect to the pair $\{\tilde{\boldsymbol{U}}_i/\boldsymbol{S_i}\}$. Due to the unknown disturbance vector $\boldsymbol{U_a}$, proposed a physically realizable control algorithm,

$$\boldsymbol{H_{i+1}} = Proj\{\bar{\boldsymbol{H}}_{\boldsymbol{i+1}}\} = Proj\{\bar{\boldsymbol{H}}^{\boldsymbol{1}}_{\boldsymbol{i+1}}, ..., \bar{\boldsymbol{H}}^{\boldsymbol{n}}_{\boldsymbol{i+1}}\}, \tag{4.13}$$

where $\bar{\boldsymbol{H}}_{\boldsymbol{i+1}} = \boldsymbol{H_i} - \beta L\boldsymbol{S_i}$ and

$$Proj\{\bar{\boldsymbol{H}}_{\boldsymbol{i+1}}\} = \begin{cases} \boldsymbol{U}^{nb} & \text{if } \bar{\boldsymbol{H}}^n_{\boldsymbol{i+1}} \geq \boldsymbol{U}^{nb} \\ -\boldsymbol{U}^{nb} & \text{if } \bar{\boldsymbol{H}}^n_{\boldsymbol{i+1}} \leq -\boldsymbol{U}^{nb} \\ \bar{\boldsymbol{H}}^n_{\boldsymbol{i+1}} & \text{otherwise} \end{cases} \tag{4.14}$$

Theorem The control scheme Eq. (4.10) with the learning rule Eq. (4.13, 4.14) converges as

(1) $\lim_{i\to\infty} V_i(t) = V(t)$

(2) $\lim_{i\to\infty} S_i(t) = 0$

where, V_i is the performance index functional

$$V_i(t) = \int_0^t \tilde{\boldsymbol{U}}_{\boldsymbol{i}}^T(\tau) L^{-1} \tilde{\boldsymbol{U}}_{\boldsymbol{i}}(\tau) d\tau \tag{4.15}$$

Proof From the definition of $\tilde{U}_i$ and the learning rule,

$$\Delta \tilde{\boldsymbol{U}}_{\boldsymbol{i}} = \tilde{\boldsymbol{U}}_{\boldsymbol{i+1}} - \tilde{\boldsymbol{U}}_{\boldsymbol{i}} = -\beta L \boldsymbol{S}_{\boldsymbol{i}} \tag{4.16}$$

Therefore,

$$V_{i+1} - V_i = \int_0^t (\tilde{\boldsymbol{U}}_{\boldsymbol{i+1}}^T L^{-1} \tilde{\boldsymbol{U}}_{\boldsymbol{i+1}} - \tilde{\boldsymbol{U}}_{\boldsymbol{i}}^T L^{-1} \tilde{\boldsymbol{U}}_{\boldsymbol{i}}) \tag{4.17}$$

$$V_{i+1} - V_i \leq \int_0^t (-2\beta \boldsymbol{S}_{\boldsymbol{i}}^T \boldsymbol{D} \dot{\boldsymbol{S}}_{\boldsymbol{i}} - \beta^2 \boldsymbol{S}_{\boldsymbol{i}}^T L \boldsymbol{S}_{\boldsymbol{i}}) d\tau \tag{4.18}$$

$$V_{i+1} - V_i \leq -2\beta \boldsymbol{S}_{\boldsymbol{i}}^T \boldsymbol{D} \boldsymbol{S}_{\boldsymbol{i}} \tag{4.19}$$

Because $\boldsymbol{S}_{\boldsymbol{i}}(0) = 0$. Hence, $V_{i+1} - V_i$ converges to zero, and $\boldsymbol{S}_{\boldsymbol{i}}$ converges to zero.

4.2.1.2 Adaptive ILC with parameter learning

$$\boldsymbol{Y}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t)) \boldsymbol{\Theta} = \boldsymbol{U} - \boldsymbol{U}_{\boldsymbol{a}} \tag{4.20}$$

In order to prove the convergence of the parameter estimator, impose the following assumption on the system.

Assumption Each element Θ^n of the parameter vector $\boldsymbol{\Theta}$ is bounded with known bound $\boldsymbol{\Theta}^{nb}$. This means that $|\boldsymbol{\Theta}^n| \leq \boldsymbol{\Theta}^{nb}$ for all $n = 1, 2, ..., l$, where l is the number

of elements of $\boldsymbol{\Theta}$.

$$\boldsymbol{E_i} = -\beta \boldsymbol{Y_i} \boldsymbol{Y_i}^T \boldsymbol{S_i} - \beta L \boldsymbol{S_i} \tag{4.21}$$

the additional term $\beta \boldsymbol{Y_i} \boldsymbol{Y_i}^T \boldsymbol{S_i}$ helps to eliminate the constrains that was imposed on β and L.

$$\boldsymbol{C_i} = -\hat{\boldsymbol{D}} \dot{\boldsymbol{e}}_{\boldsymbol{i}} \tag{4.22}$$

where $\hat{\boldsymbol{D}}$ denotes the estimated variables. Substitute Eq. (4.21) and (4.22) into Eq. (4.4),

$$\boldsymbol{D} \dot{\boldsymbol{S}}_{\boldsymbol{i}} + \beta L \boldsymbol{S_i} + \beta \boldsymbol{Y_i} \boldsymbol{Y_i}^T \boldsymbol{S_i} = \boldsymbol{Y_i} \tilde{\boldsymbol{\Theta}} + \tilde{\boldsymbol{U}}_{\boldsymbol{i}} \tag{4.23}$$

$$\tilde{\boldsymbol{\Theta}}_{\boldsymbol{i}} = \boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}_{\boldsymbol{i}} \tag{4.24}$$

In order to construct good learning rules, define $W(\boldsymbol{S_i}) = \frac{1}{2} \boldsymbol{S_i}^T \boldsymbol{D} \boldsymbol{S_i}$ So that,

$$W(\boldsymbol{S_i}(t)) - W(\boldsymbol{S_i}(0)) = -\beta \int_0^t \boldsymbol{S}_i^T (L + \boldsymbol{Y_i} \boldsymbol{Y_i}^T) \boldsymbol{S_i} d\tau + \int_0^t \boldsymbol{S_i}^T (\tilde{\boldsymbol{U}}_{\boldsymbol{i}} + \boldsymbol{Y_i} \tilde{\boldsymbol{U}}_{\boldsymbol{i}}) d\tau \tag{4.25}$$

Proposing an adaptive learning controller with following parameter learning rule:

$$\hat{\boldsymbol{\Theta}}_{\boldsymbol{i+1}} = Proj\{\hat{\bar{\boldsymbol{\Theta}}}_{\boldsymbol{i+1}}\} = Proj\{\hat{\bar{\boldsymbol{\Theta}}}^1_{\boldsymbol{i+1}}, ..., \hat{\bar{\boldsymbol{\Theta}}}^n_{\boldsymbol{i+1}}\}, \tag{4.26}$$

where $\hat{\bar{\boldsymbol{\Theta}}}_{\boldsymbol{i+1}} = \hat{\boldsymbol{\Theta}}_{\boldsymbol{i}} - \beta \boldsymbol{Y_i} \boldsymbol{S_i}$ and

$$Proj\{\hat{\bar{\boldsymbol{\Theta}}}_{\boldsymbol{i+1}}\} = \begin{cases} \boldsymbol{\Theta}^{nb} & \text{if } \hat{\bar{\boldsymbol{\Theta}}}^n_{\boldsymbol{i+1}} \leq \boldsymbol{\Theta}^{nb} \\ -\boldsymbol{\Theta}^{nb} & \text{if } \hat{\bar{\boldsymbol{\Theta}}}^n_{\boldsymbol{i+1}} \leq -\boldsymbol{\Theta}^{nb} \\ \hat{\bar{\boldsymbol{\Theta}}}^n_{\boldsymbol{i+1}} & \text{otherwise} \end{cases} \tag{4.27}$$

The designed adaptive learning controller is composed of the feedback controller, the learning control rule and the parameter leaning rule, which is shown below,

$$\boldsymbol{U} = \boldsymbol{E_i} + \boldsymbol{H_i} + \boldsymbol{Y_i}\hat{\boldsymbol{\Theta}}_{\boldsymbol{i}}; \tag{4.28}$$

Theorem The adaptive learning controller for the uncertain dynamic system converges uniformly as follows

(1) $\lim_{i\to\infty} V_i(t) = V(t)$

(2) $\lim_{i\to\infty} S_i(t) = 0$

where, V_i is the performance index functional

$$V_i(t) = \int_0^t (\tilde{\boldsymbol{U}}_{\boldsymbol{i}}^T(\tau) L^{-1} \tilde{\boldsymbol{U}}_{\boldsymbol{i}}(\tau) + \tilde{\boldsymbol{\Theta}}_{\boldsymbol{i}}^T \tilde{\boldsymbol{\Theta}}_{\boldsymbol{i}}) d\tau \tag{4.29}$$

Proof Since $\mid \tilde{\bar{\boldsymbol{U}}}_{\boldsymbol{i}} \mid \geq \mid \tilde{\boldsymbol{U}}_{\boldsymbol{i}} \mid$ and $\mid \tilde{\bar{\boldsymbol{\Theta}}}_{\boldsymbol{i}} \mid \geq \mid \tilde{\boldsymbol{\Theta}}_{\boldsymbol{i}} \mid$ so that

$$V_{i+1} - V_i \leq \bar{V}_{i+1} - V_i \tag{4.30}$$

where,

$$\bar{V}_i = \int_0^t (\tilde{\bar{\boldsymbol{U}}}_{\boldsymbol{i}}^T(\tau) L^{-1} \tilde{\bar{\boldsymbol{U}}}_{\boldsymbol{i}}(\tau) + \tilde{\bar{\boldsymbol{\Theta}}}_{\boldsymbol{i}}^T(\tau) \tilde{\bar{\boldsymbol{\Theta}}}_{\boldsymbol{i}}(\tau)) \tag{4.31}$$

$$\Delta\tilde{\boldsymbol{\Theta}}_{\boldsymbol{i}} = \hat{\bar{\boldsymbol{\Theta}}}_{\boldsymbol{i+1}} - \hat{\boldsymbol{\Theta}}_{\boldsymbol{i}} = -\beta \boldsymbol{Y_i} \boldsymbol{S_i} \tag{4.32}$$

and

$$V_{i+1}(t) - V_i(t) \leq -2\beta \int_0^t \boldsymbol{S}_{\boldsymbol{i}}^T \boldsymbol{D} \dot{\boldsymbol{S}}_{\boldsymbol{i}} d\tau - \beta^2 \int_0^T \boldsymbol{S}_{\boldsymbol{i}}^T (L + \beta \boldsymbol{Y_i} \boldsymbol{Y_i}^T) \boldsymbol{S_i} d\tau \tag{4.33}$$

$$V_{i+1}(t) - V_i(t) \leq -\beta \boldsymbol{S_i^T D S_i} \tag{4.34}$$

Because $\boldsymbol{S_i}(0) = 0$, Hence, $V_{i+1} - V_i$ converges to zero, and $\boldsymbol{S_i}$ converges to zero.

4.2.2 Controller Design Based on the Position Control

The position control considers the position of the UAV, which is Z, X and Y, meanwhile, keeps yaw direction the same as the previous one. In this case, ϕ and θ are considered to be U_2 and U_3, which are the control input.

4.2.2.1 Learning control without parameter adaptation

$U = [U_1\ \phi\ \theta\ U_4]'$ is the control input.

$$U = E_i + C_i + H_i \tag{4.35}$$

where $E_i = -\beta LS$ is the feedback control input, $C_i = D\dot{e}_i$ is the nonlinear compensation term. Defined four sliding surfaces shown as

$$s_1 = (z_i - z_d) + (\dot{z}_i - \dot{z}_d) \tag{4.36}$$

$$s_2 = (y_i - y_d) + (\dot{y}_i - \dot{y}_d) \tag{4.37}$$

$$s_3 = (x_i - x_d) + (\dot{x}_i - \dot{x}_d) \tag{4.38}$$

$$s_4 = (\psi_i - \psi_d) + (\dot{\psi}_i - \dot{\psi}_d) \tag{4.39}$$

Based on Eqs. (4.36)-(4.39), could have $S = [s_1\ s_2\ s_3\ s_4]'$ and $\dot{S} = [\dot{s}_1\ \dot{s}_2\ \dot{s}_3\ \dot{s}_4]'$, $\dot{S}$ expressed as

$$\dot{s}_1 = (\dot{z}_i - \dot{z}_d) + (\ddot{z}_i - \ddot{z}_d) \tag{4.40}$$

$$\dot{s}_2 = (\dot{y}_i - \dot{y}_d) + (\ddot{y}_i - \ddot{y}_d) \tag{4.41}$$

$$\dot{s}_3 = (\dot{x}_i - \dot{x}_d) + (\ddot{x}_i - \ddot{x}_d) \tag{4.42}$$

$$\dot{s}_4 = (\dot{\psi}_i - \dot{\psi}_d) + (\ddot{\psi}_i - \ddot{\psi}_d) \tag{4.43}$$

Substitute Eq.(4.35) into Eq.(4.4), could have

$$D\ddot{q}_i + U_a = E_i + C_i + H_i \tag{4.44}$$

$$D(\ddot{e}_i + \ddot{q}_d) + U_a = -\beta LS + D\dot{e}_i + H_i \tag{4.45}$$

Therefore,

$$D\dot{s} + \beta LS = -(D\ddot{q}_d + U_a - H_i) = -(U_d^* + U_a - H_i) = -(U_d - H_i) = \tilde{U}_i \tag{4.46}$$

To generate the learning algorithm, define a Lyapunov function candidate $W(S_i) = \frac{1}{2}S_i^T D S_i$, then, the derivative of $W(S_i)$ along the error trajectory is

$$\dot{W}(S_i) = S_i^T D\dot{S}_i + \frac{1}{2}S_i^T \dot{D} S_i = -\beta S_i^T L S_i + S_i^T \tilde{U}_i \tag{4.47}$$

Integrating both sides of the above equation, could obtain

$$\dot{W}(S_i(t)) - \dot{W}(S_i(0)) = -\int_0^t \beta S_i^T L S_i d\tau + \int_0^t S_i^T \tilde{U}_i d\tau \tag{4.48}$$

which indicates that the error dynamics is strictly passive with respect to the pair $\{\tilde{U}_i/S_i\}$. Due to unknown disturbance vector U_d, proposed a physically realizable control algorithm,

$$H_{i+1} = Proj\{\bar{H}_{i+1}\} = Proj\{\bar{H}^1_{i+1}, ..., \bar{H}^n_{i+1}\}, \tag{4.49}$$

where $\bar{H}_{i+1} = H_i - \beta L S_i$ and

$$Proj\{\bar{H}_{i+1}\} = \begin{cases} U^{nb} & \text{if } \bar{H}^n_{i+1} \geq U^{nb} \\ -U^{nb} & \text{if } \bar{H}^n_{i+1} \leq -U^{nb} \\ \bar{H}^n_{i+1} & \text{otherwise} \end{cases} \tag{4.50}$$

Theorem The control scheme Eq. (4.46) with the learning rule Eq. (4.49, 4.50) converges as

(1) $\lim_{i\to\infty} V_i(t) = V(t)$

(2) $\lim_{i\to\infty} S_i(t) = 0$

where V_i is the performance index functional

$$V_i(t) = \int_0^t \tilde{U}_i^T(\tau) L^{-1} \tilde{U}_i(\tau) d\tau \tag{4.51}$$

Proof From the definition of $\tilde{U}_i$ and the learning rule,

$$\Delta\tilde{U}_i = \tilde{U}_{i+1} - \tilde{U}_i = -\beta L S_i \tag{4.52}$$

Therefore,

$$V_{i+1} - V_i = \int_0^t (\tilde{U}_{i+1}^T L^{-1} \tilde{U}_{i+1} - \tilde{U}_i^T L^{-1} \tilde{U}_i) \tag{4.53}$$

$$V_{i+1} - V_i \leq \int_0^t (-2\beta S_i^T D \dot{S}_i - \beta^2 S_i^T L S_i) d\tau \tag{4.54}$$

$$V_{i+1} - V_i \leq -2\beta S_i^T D S_i \tag{4.55}$$

Because $S_i(0) = 0$. Hence, $V_{i+1} - V_i$ converges to zero, and S_i converges to zero.

4.2.2.2 Adaptive ILC with parameter learning

$$Y(q(t), \dot{q}(t), \ddot{q}(t))\Theta = U - U_a \tag{4.56}$$

In order to prove the convergence of the parameter estimator, impose the following assumption on system.

Assumption Each element Θ^n of the parameter vector Θ is bounded with known bound Θ^{nb}. This means that $|\Theta_i| \leq \Theta^{nb}$ for all $n = 1, 2, \cdots, l$, where l is the number of elements of Θ.

$$E_i = -\beta Y_i Y_i^T S_i - \beta L S_i \tag{4.57}$$

The additional term $\beta Y_i Y_i^T S_i$ helps to eliminate the constrains that was imposed on β and L.

$$C_i = -\hat{D}\dot{e}_i \tag{4.58}$$

where $\hat{\cdot}$ denotes the estimated variables. Substitute Eq. (4.57) and (4.58) into Eq. (4.4),

$$D\dot{S}_i + \beta L S_i + \beta Y_i Y_i^T S_i = Y_i \tilde{\Theta} + \tilde{U}_i \tag{4.59}$$

$$\tilde{\Theta}_i = \Theta - \hat{\Theta}_i \tag{4.60}$$

In order to construct good learning rules, define $W(S_i) = \frac{1}{2} S_i^T D S_i$ So that,

$$W(S_i(t)) - W(S_i(0)) = -\beta \int_0^t S_i^T (L + Y_i Y_i^T) S_i d\tau + \int_0^t S_i^T (\tilde{U}_i + Y_i \tilde{U}_i) d\tau \tag{4.61}$$

Proposed an adaptive learning controller with following parameter learning rule:

$$\hat{\Theta}_{i+1} = Proj\{\hat{\bar{\Theta}}_{i+1}\} = Proj\{\hat{\bar{\Theta}}^1_{i+1}, \cdots, \hat{\bar{\Theta}}^n_{i+1}\}, \tag{4.62}$$

where $\hat{\bar{\Theta}}_{i+1} = \hat{\Theta}_i - \beta Y_i S_i$ and

$$Proj\{\hat{\bar{\Theta}}_{i+1}\} = \begin{cases} \Theta^{nb} & \text{if } \hat{\bar{\Theta}}^n_{i+1} \leq \Theta^{nb} \\ -\Theta^{nb} & \text{if } \hat{\bar{\Theta}}^n_{i+1} \leq -\Theta^{nb} \\ \hat{\bar{\Theta}}^n_{i+1} & \text{otherwise} \end{cases} \tag{4.63}$$

The designed adaptive learning controller be composed of the feedback controller, the learning control rule and parameter leaning rule, which is shown below,

$$U = E_i + H_i + Y_i \hat{\Theta}_i; \tag{4.64}$$

Theorem The adaptive learning controller for the uncertain dynamic system converges uniformly as follows

(1) $\lim_{i\to\infty} V_i(t) = V(t)$

(2) $\lim_{i\to\infty} S_i(t) = 0$

where V_i is the performance index functional

$$V_i(t) = \int_0^t (\tilde{U}_i^T(\tau)L^{-1}\tilde{U}_i(\tau) + \tilde{\Theta}_i^T\tilde{\Theta}_i)d\tau \tag{4.65}$$

Proof Since $\mid \tilde{\tilde{U}}_i \mid \geq \mid \tilde{U}_i \mid$ and $\mid \tilde{\tilde{\Theta}}_i \mid \geq \mid \tilde{\Theta}_i \mid$ so that

$$V_{i+1} - V_i \leq \bar{V}_{i+1} - V_i \tag{4.66}$$

where

$$\bar{V}_i = \int_0^t (\tilde{\tilde{U}}_i^T(\tau)L^{-1}\tilde{\tilde{U}}_i(\tau) + \tilde{\tilde{\Theta}}_i^T(\tau)\tilde{\tilde{\Theta}}_i(\tau)) \tag{4.67}$$

$$\Delta\tilde{\Theta}_i = \hat{\tilde{\Theta}}_{i+1} - \hat{\Theta}_i = -\beta Y_i S_i \tag{4.68}$$

and

$$V_{i+1}(t) - V_i(t) \leq -2\beta \int_0^t S_i^T D\dot{S}_i d\tau - \beta^2 \int_0^T S_i^T(L + \beta Y_i Y_i^T)S_i d\tau \tag{4.69}$$

$$V_{i+1}(t) - V_i(t) \leq -\beta S_i^T D S_i \tag{4.70}$$

Because $S_i(0) = 0$, Hence, $V_{i+1} - V_i$ converges to zero, and S_i converges to zero.

4.3 Simulation Results

This section presents the trajectory tracking simulation results of the UAV, using the proposed adaptive-type ILCs. All the simulation results are obtained solely

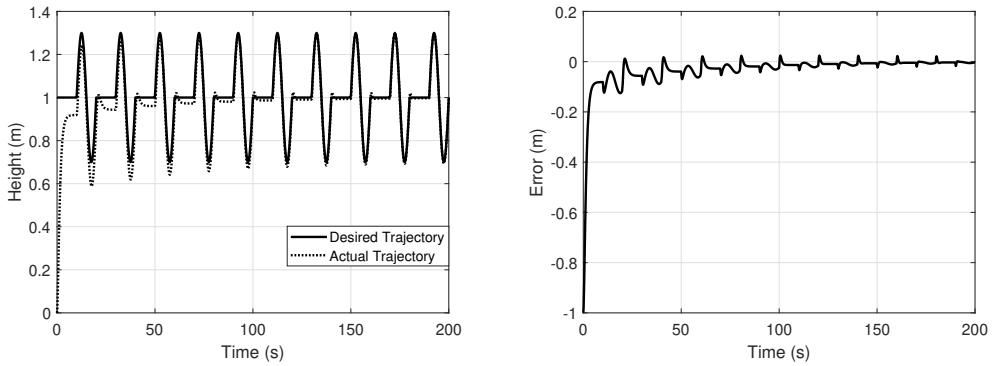
using MATLAB & Simulink programming.

Translation in Z-axis Direction

Throughout the simulation, the iteration time is 20 seconds and desired trajectory in Z is given as

$$z_d = \begin{cases} 1 & \text{if } 0 \leq T \leq 10 \\ 0.3\sin(\frac{\pi}{5}T) - 1 & \text{if } 10 < T \leq 20 \end{cases} \tag{4.71}$$

where, $T \in (0, 20]$ is the time in each iteration, the mass of the UAV used in the simulation is $m = 1.12$ kg and the acceleration of gravity is $g = 9.81$ m/s^2. The initial parameters for the proposed controller are estimated as $s_{z1}(T) = 0$, $H_{z1}(T) = 0$, $\widehat{\Theta}_{z1}(T) = 1.1$ and $Y_{z1}(T) = 0$. As shown in Fig. 4.6, the actual trajectory gradually tracks the desired trajectory and the tracking error decreases.



(a) Tracking performance (b) Tracking error

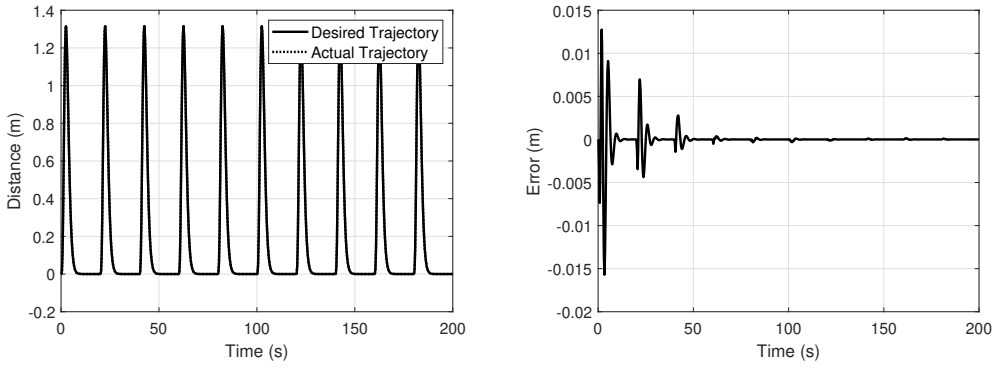
Figure 4.6: QDrone simulation results : trajectory tracking in Z-axis direction

Translation in X-axis Direction

The desired trajectory on X direction used in the simulation is

$$x_d = RT^5 e^{(-2T)} \tag{4.72}$$

where, $R = 2$, and T is the time in each iteration. The proposed controller drives the UAV to track the desired trajectory in X-axis direction, and the tracking error shows the learning performance iteration by iteration, which is described as Fig. 4.7.

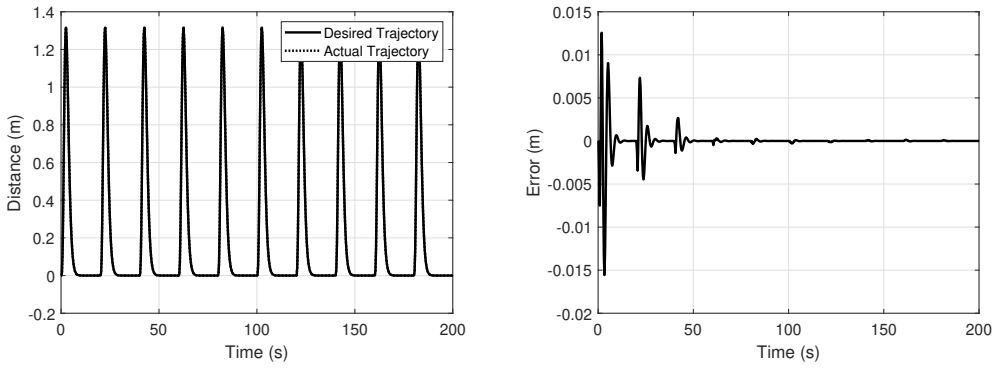


(a) Tracking performance (b) Tracking error

Figure 4.7: QDrone simulation results : trajectory tracking in X-axis direction

Translation in Y-axis Direction

The desired trajectory and all parameters are setting as in the translation in X-axis direction. The tracking performance is illustrated as Fig. 4.8.

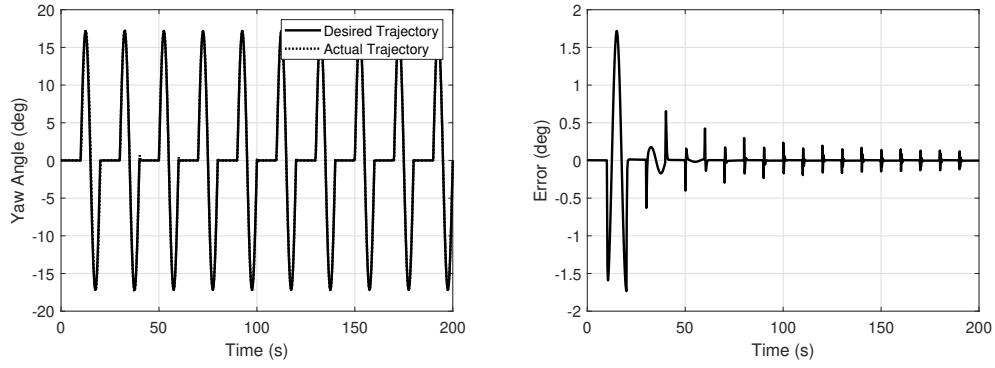


(a) Tracking performance (b) Tracking error

Figure 4.8: QDrone simulation results : trajectory tracking in Y-axis direction

Rotation in Yaw Direction

The given trajectory is the same as the trajectory in Z-axis direction, and $I_z = 0.002$ kg/m^2. From Fig. 4.9, it can be seen that the tracking performance is promoted as the number of iterations increased.

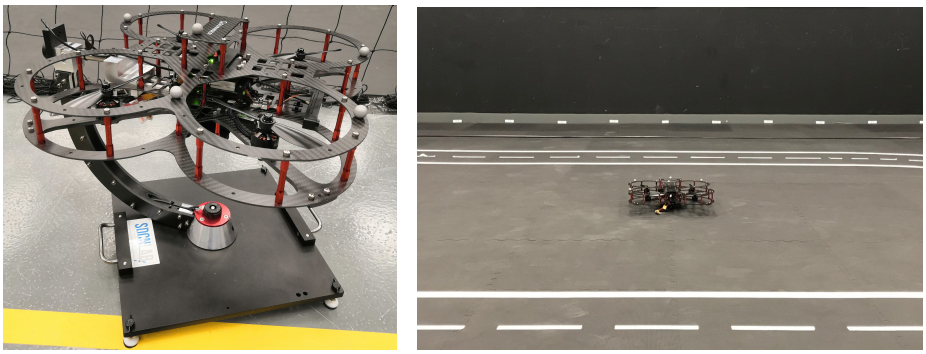


(a) Tracking performance (b) Tracking error

Figure 4.9: QDrone simulation results : trajectory tracking in Yaw direction

4.4 Experimental Results

This section presents the experimental results by using the QDrone [75], with the proposed adaptive-type ILCs. The experimental results are divided into two parts: the test with the gimbal and the free flight test.



(a) The gimbal and the QDrone (b) The QDrone in flying space

Figure 4.10: QDrone experimental system

4.4.1 Experimental Results with Gimbal

The gimbal is a frame that the QDrone can be mounted on it. In this case, there are two degrees of freedom, i.e, roll and yaw. Fig. 4.10(a) shows the QDrone with gimbal. To do the test, first, mount the QDrone on the gimbal, then give the desired trajectory in directions such as roll or yaw. While the QDrone is mounted on the gimbal, the dynamic model of the QDrone is considered to be the same as in free flight test, since the motion is equivalent to an attitude motion of a QDrone

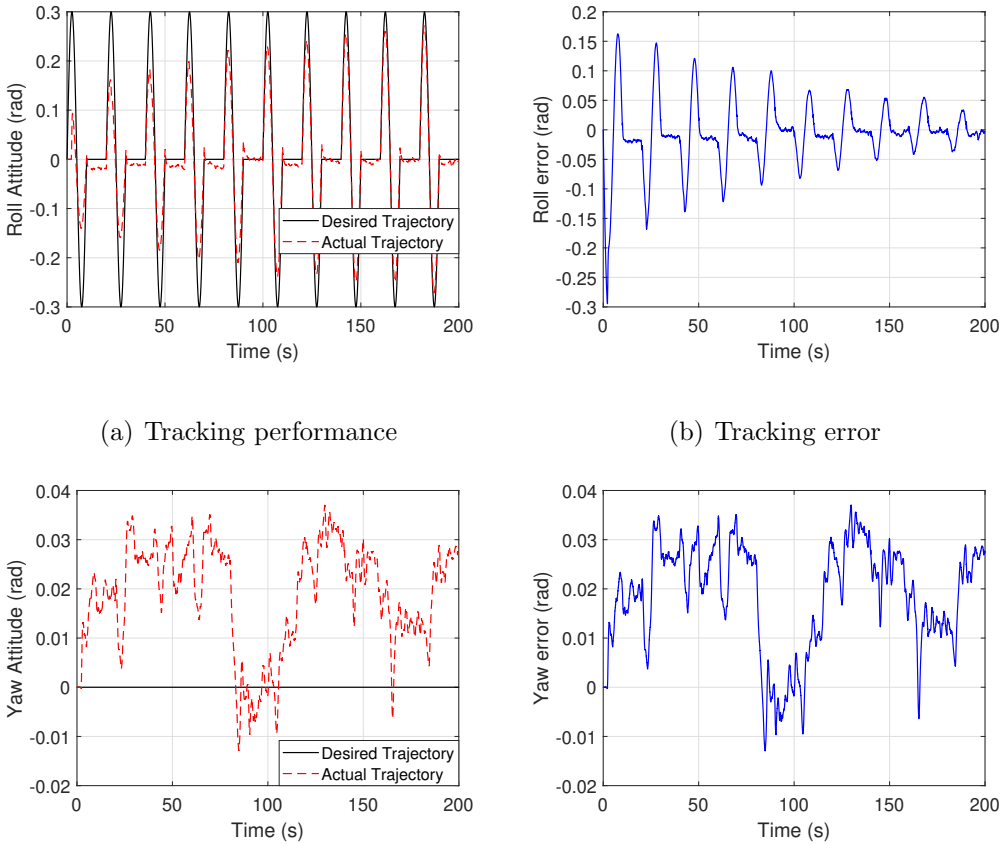
in hovering. It is worth nothing that the gimbal has no actuators and the QDrone is actuated only by the four propellers.

Rotation in the Roll Direction

The given trajectory in roll direction is

$$\phi_d = \begin{cases} 0.3\sin(\frac{\pi}{10}T) & \text{if } 0 < T \leq 10 \\ 0 & \text{if } 10 \leq T \leq 20 \end{cases} \tag{4.73}$$

where, $T \in (0, 20]$ is time for each iteration. The yaw angle is set to zero. As shown in Fig. 4.11, the tracking performance is promoted as the time increases, meanwhile, the yaw direction is stable.



(a) Tracking performance

(b) Tracking error

(c) Tracking performance

(d) Tracking error

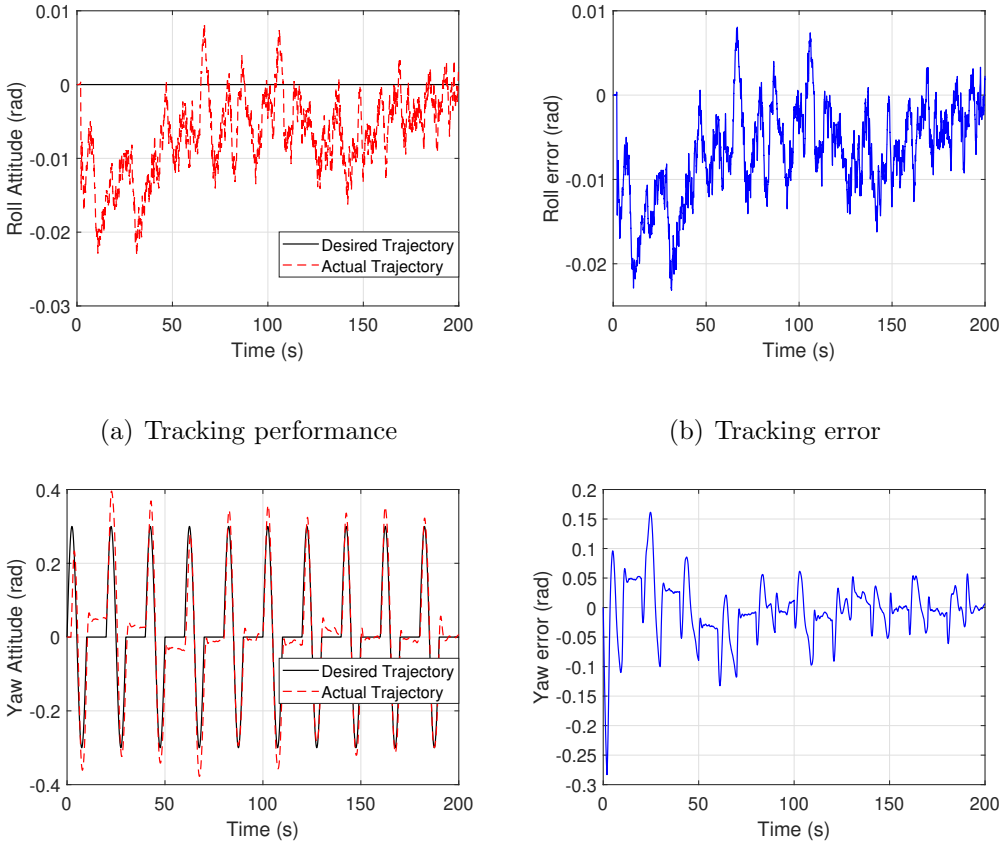
Figure 4.11: Experimental results with the Gimbal : rotation in the roll angle & yaw angle maintains stable

Rotation in the Yaw Direction

Similarly, the given trajectory in the yaw direction is

$$\psi_d = \begin{cases} 0.3\sin(\frac{\pi}{10}T) & \text{if } 0 < T \leq 10 \\ 0 & \text{if } 10 \leq T \leq 20 \end{cases} \tag{4.74}$$

where, $T \in (0, 20]$ is time for each iteration, conversely, the roll angel is set to zero. As shown in Fig. 4.12, the tracking performance is promoted as the time increases, meanwhile, the roll direction is stable.



(a) Tracking performance

(b) Tracking error

(c) Tracking performance

(d) Tracking error

Figure 4.12: Experimental results with the Gimbal : rotation in the yaw angle & roll angle maintains stable

4.4.2 Free Flight Test

The tests have been run in the York University Autonomous Unmanned Vehicle (YU-AUV) facility in the Spacecraft Dynamics Control and Navigation Lab (SDC-NLab). The QDrone has 5 markers on the top of it with a unique shape, and there are 16 cameras on the wall to recognize the real-time position of the QDrone, as the shown in Fig. 4.10(b). For all the tests, the desired trajectories are given after the QDrone takes off. But before that, the system considers the initial position of the QDrone on the map is the desired one, therefore, the tracking error is zero during that period. And the precision of this experimental system is 0.05 m.

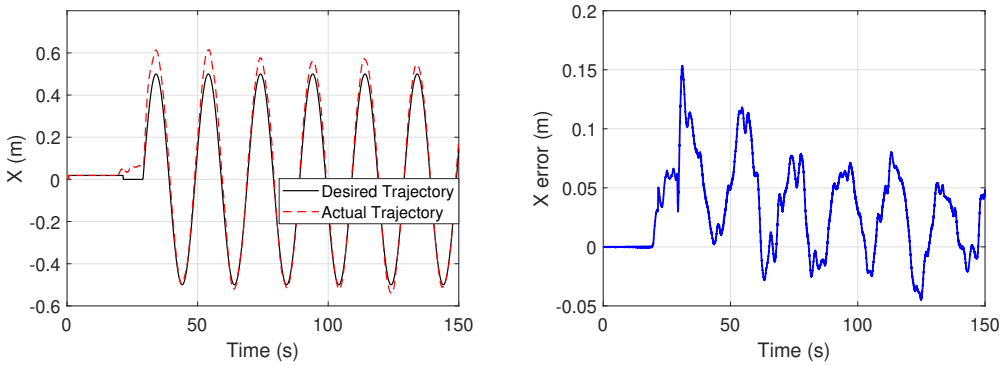
Translation in the X-axis Direction

First, the motion is only in the X-axis direction, and the motion in the other directions are set to zero. The desired trajectory is

$$x_d = 0.5\sin(\frac{\pi}{10}T) \tag{4.75}$$

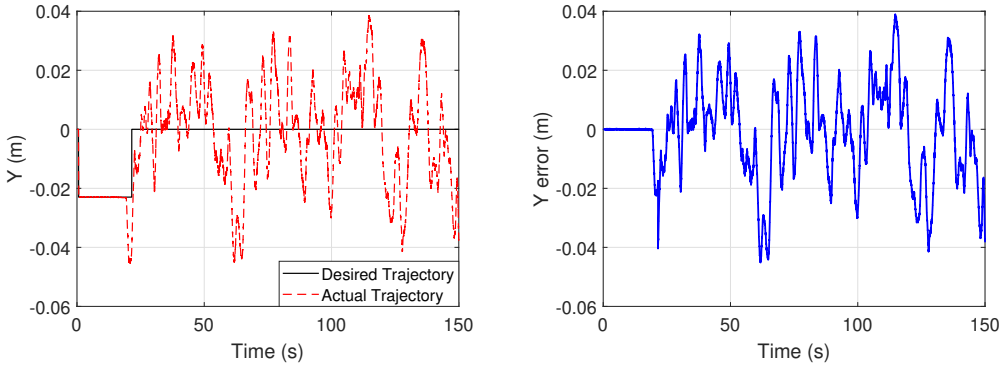
where, $T \in (0, 20]$ is time for each iteration. As shown in Fig. 4.13, the tracking performance is promoted as the time increases. At the 1^{st} iteration, the tracking error is 0.15 m, but at the 6^{th} iteration it drops to 0.05 m. From Fig. 4.14, it can be seen that the QDrone is stable in the Y-axis direction since the tracking error is within ± 0.05 m. Also, in the Z-axis direction, the learning effect is shown from

Fig. 4.15. Meanwhile, the tracking error in the Yaw-axis direction is around ±0.04 rad as shown in Fig. 4.16.



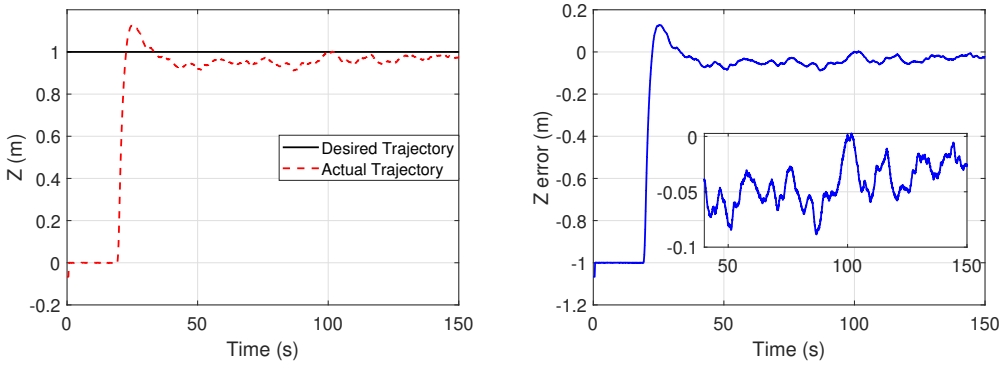
(a) Tracking performance (b) Tracking error

Figure 4.13: Free flight test - translation in the X-axis direction only : results in the X-axis direction



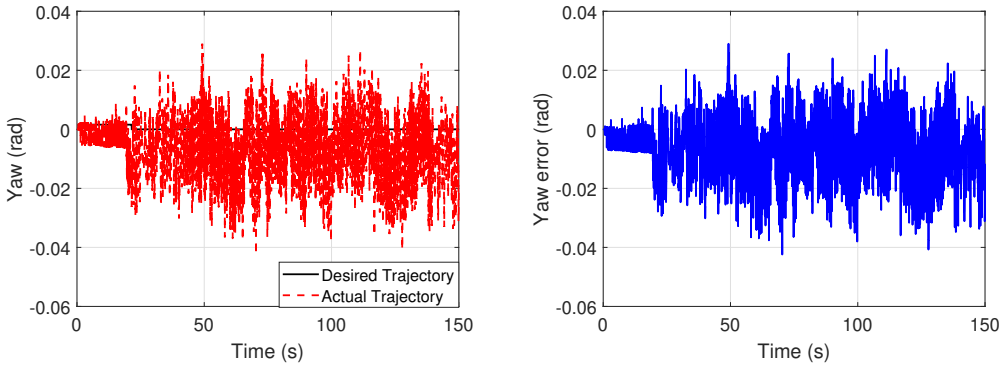
(a) Tracking performance (b) Tracking error

Figure 4.14: Free flight test - translation in the X-axis direction only : results in the Y-axis direction



(a) Tracking performance (b) Tracking error

Figure 4.15: Free flight test - translation in the X-axis direction only : results in the Z-axis direction



(a) Tracking performance (b) Tracking error

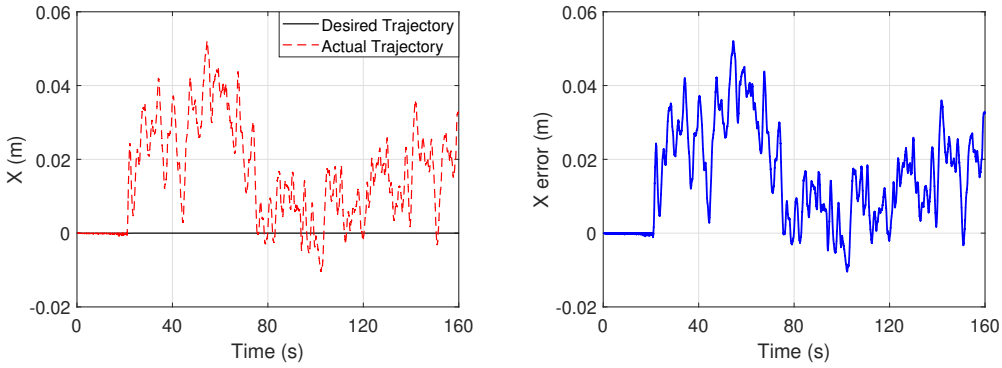
Figure 4.16: Free flight test - translation in the X-axis direction only : results in the Yaw angle

Translation in the Y-axis Direction

The trajectory in the Y-axis direction is the same as in the X-axis direction, as shown below,

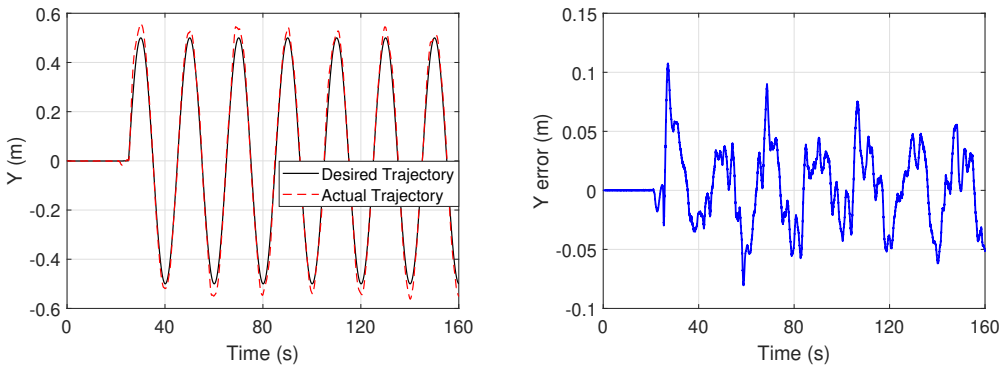
$$y_d = 0.5 \sin(\frac{\pi}{10} T) \tag{4.76}$$

where, $T \in (0, 20]$ is time for each iteration. In this test, there is no desired motion in the X-axis direction. According to Fig. 4.17, the QDrone is stable in the X-axis direction since the tracking error is between -0.02 m and 0.06 m. At the same time, Fig. 4.18 illustrates the learning performance in the Y-axis direction, since the tracking error declines from more than 0.1 m to ± 0.05 m as iteration time added. Similarly, Fig. 4.19 shows the learning performance in the Z-axis direction. The actual altitude converges to the desired one, which is 1 m, and the error in the Yaw-axis direction is around ± 0.05 rad.



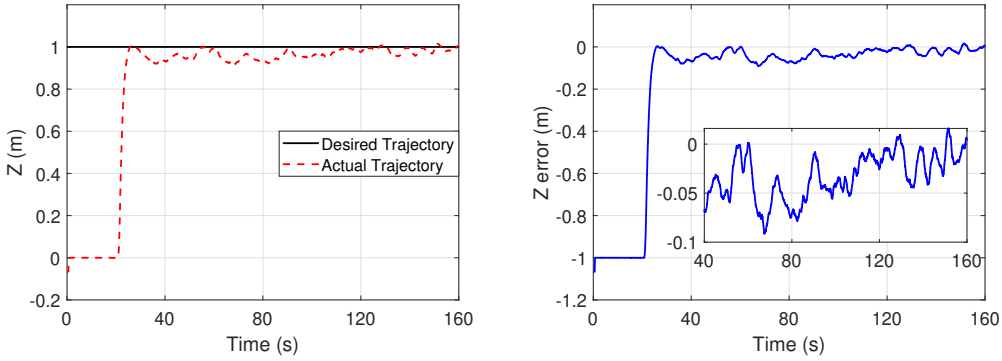
(a) Tracking performance (b) Tracking error

Figure 4.17: Free flight test - translation in the Y-axis direction only : results in the X-axis direction



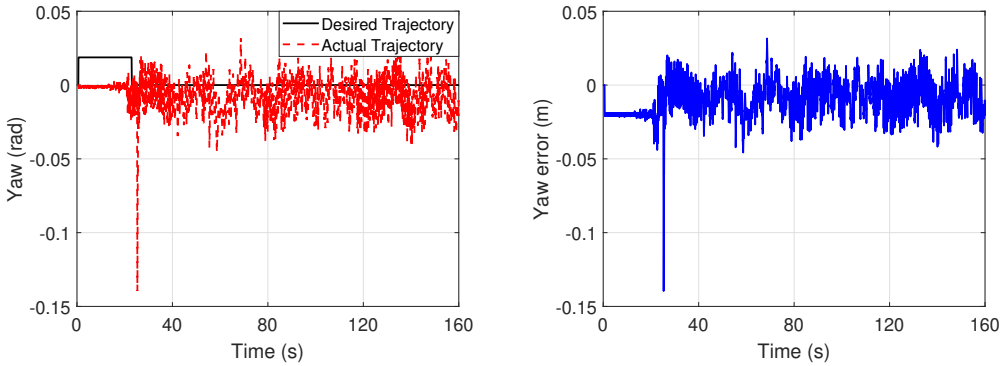
(a) Tracking performance (b) Tracking error

Figure 4.18: Free flight test - translation in the Y-axis direction only : results in the Y-axis direction



(a) Tracking performance (b) Tracking error

Figure 4.19: Free flight test - translation in the Y-axis direction only : results in the Z-axis direction



(a) Tracking performance (b) Tracking error

Figure 4.20: Free flight test - translation in the Y-axis direction only : results in the Yaw angle

Translation in the Z-axis Direction

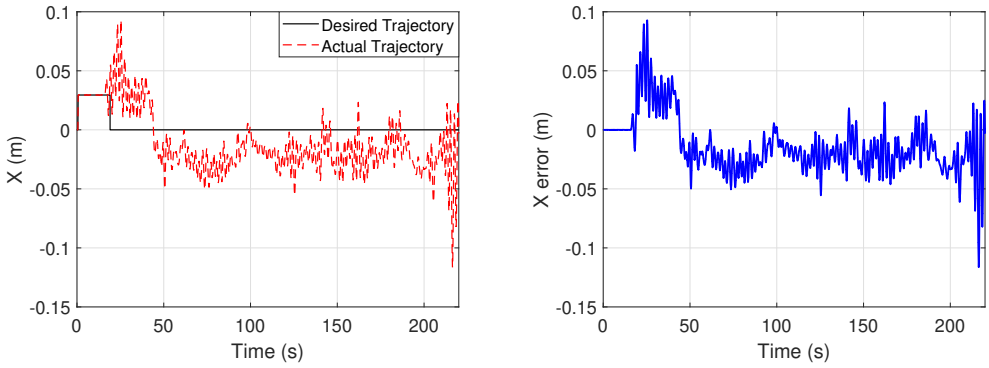
In this test, there are three different desired trajectory in the Z-axis direction, which are used to verify the present controller drive the QDrone to track a variety of trajectories. The attached experimental results are all illustrate the QDrone improves the flying behavior in the Z-axis direction, and maintain no motion in the other directions.

Test 1

The trajectory in the Z-axis direction is

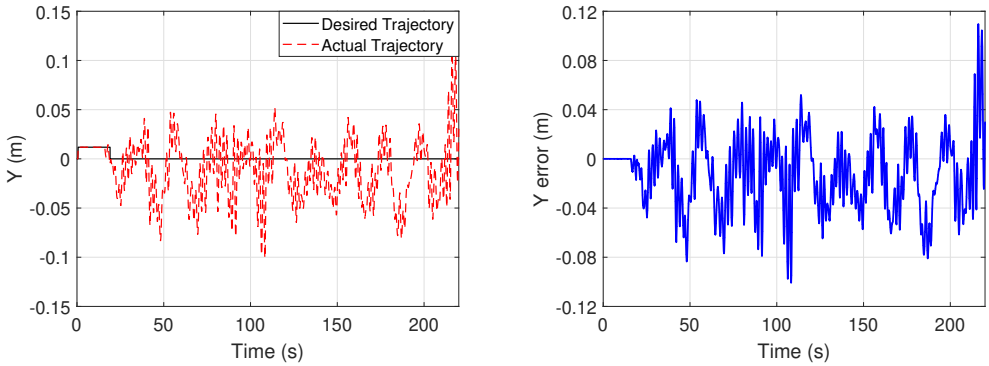
$$z_d = \begin{cases} 0.8 + 0.3\sin(\frac{\pi}{10}T) & \text{if } 0 < T \leq 10 \\ 0.8 & \text{if } 10 \leq T \leq 20 \end{cases} \tag{4.77}$$

where $T \in (0, 20]$ is time for each iteration. Figs. 4.21 and 4.22 describe the QDrone maintains no motion, since the tracking error are all acceptable. Meanwhile, the actual trajectory of the QDrone converges to the desired one as time increases, and the tracking error converge to around 0 m, as shown in Fig. 4.23. Fig. 4.24 shows the tracking error in the Yaw angle is between 0.06 rad and -0.08 rad.



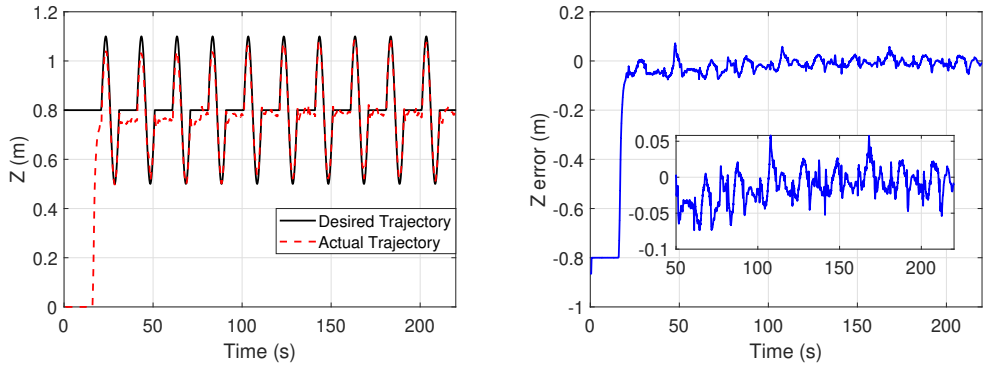
(a) Tracking performance (b) Tracking error

Figure 4.21: Free flight test - translation in the Z-axis direction only : results in the X-axis direction - test 1



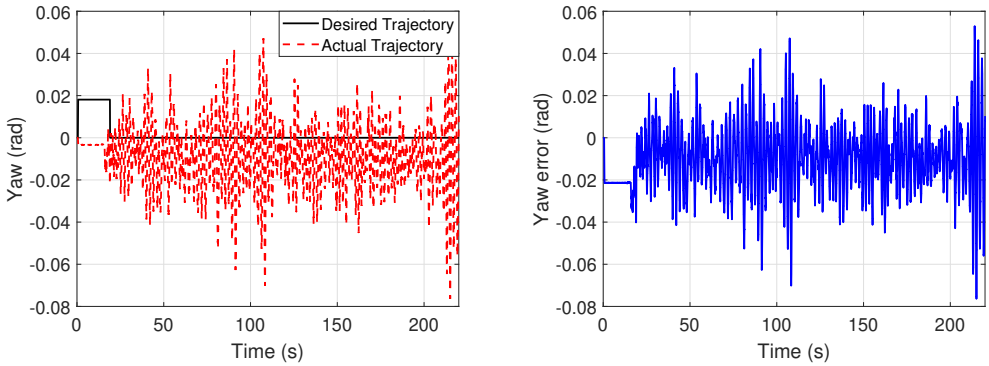
(a) Tracking performance (b) Tracking error

Figure 4.22: Free flight test - translation in the Z-axis direction only : results in the Y-axis direction - test 1



(a) Tracking performance (b) Tracking error

Figure 4.23: Free flight test - translation in the Z-axis direction only : results in the Z-axis direction - test 1



(a) Tracking performance (b) Tracking error

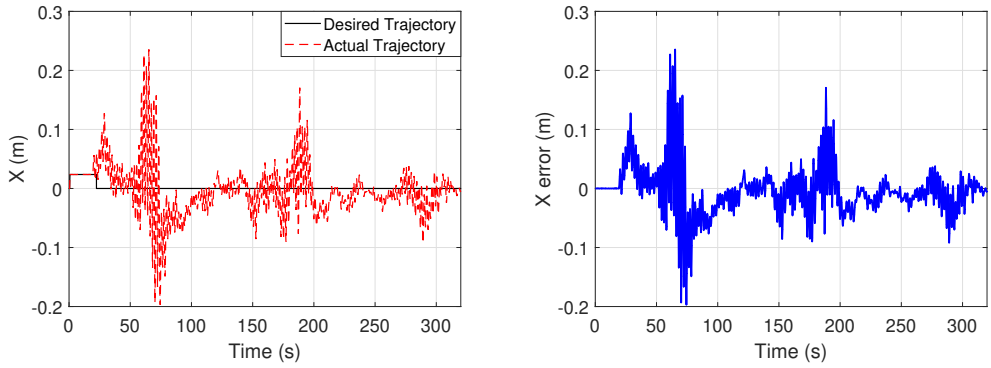
Figure 4.24: Free flight test - translation in the Z-axis direction only : results in the Yaw angle - test 1

Test 2

The trajectory in the Z-axis direction is

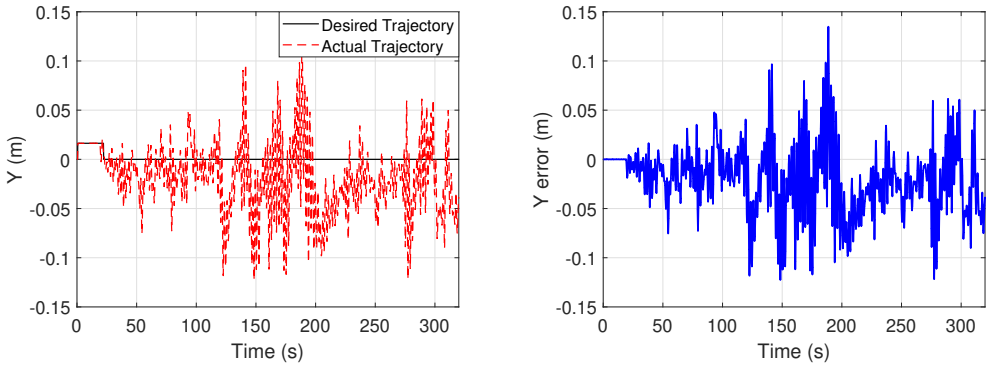
$$z_d = 0.8 + 0.3\sin(\frac{\pi}{10}T) \tag{4.78}$$

where, $T \in (0, 20]$ is time for each iteration. According to Fig. 4.25, the QDrone improves the tracking behavior in the X-axis direction, the tracking error reduced from more than ± 0.2 m to 0.1 m. At the same time, Fig. 4.26 shows the QDrone is stable In the Y-axis direction. Fig. 4.27 provides the learning performance in the Z-axis direction, since there is a tendency that the tracking error goes to 0 m, and the error in the Yaw angle is within ± 0.1 rad.



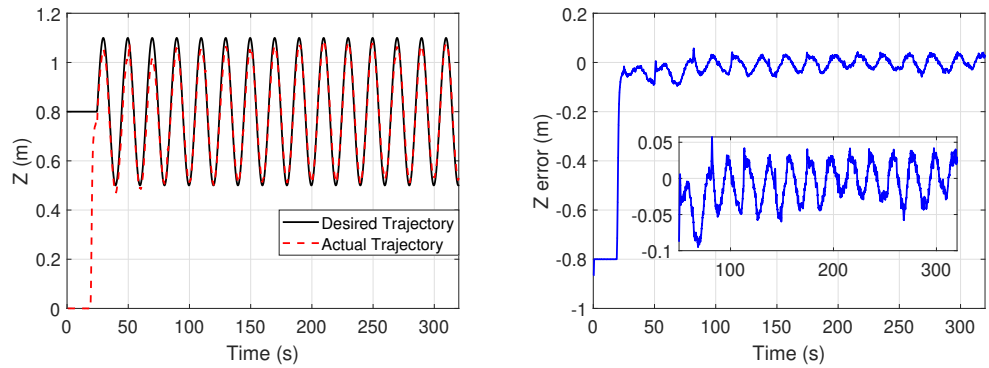
(a) Tracking performance (b) Tracking error

Figure 4.25: Free flight test - translation in the Z-axis direction only : results in the X-axis direction - test 2



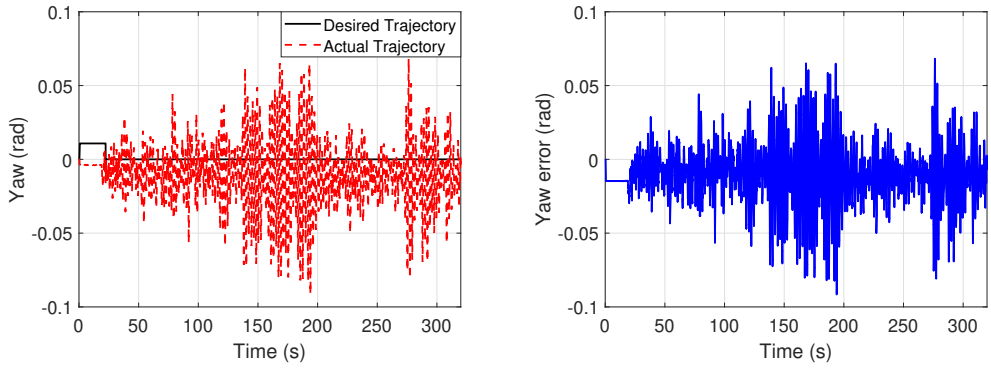
(a) Tracking performance (b) Tracking error

Figure 4.26: Free flight test - translation in the Z-axis direction only : results in the Y-axis direction - test 2



(a) Tracking performance (b) Tracking error

Figure 4.27: Free flight test - translation in the Z-axis direction only : results in the Z-axis direction - test 2



(a) Tracking performance (b) Tracking error

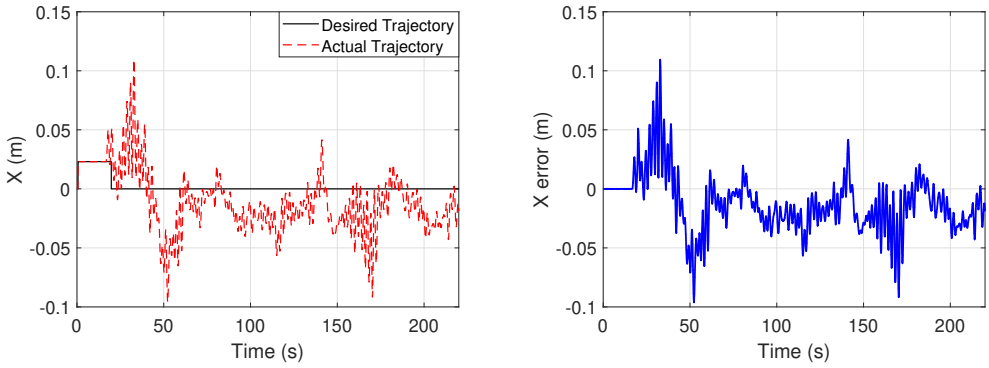
Figure 4.28: Free flight test - translation in the Z-axis direction only : results in the Yaw angle - test 2

Test 3

The trajectory in the Z-axis direction is

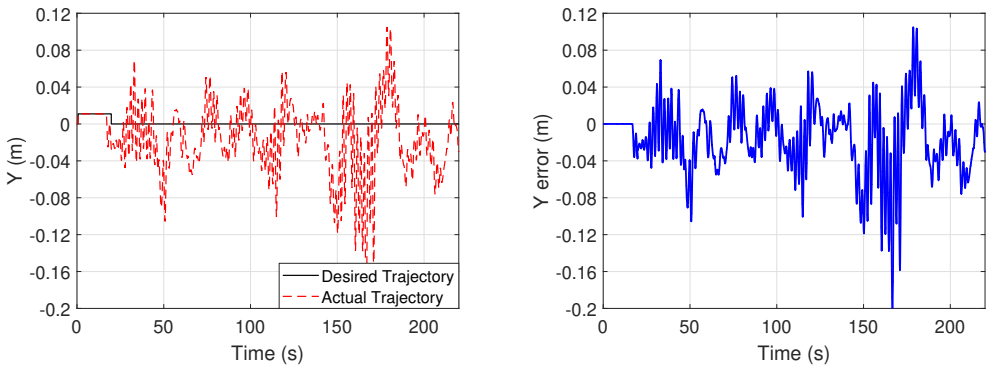
$$z_d = 0.8 + 0.3\sin(\frac{\pi}{5}T) \tag{4.79}$$

where, $T \in (0, 10]$ is time for each iteration. Similarly, Figs. 4.29 and 4.30 describe the QDrone is stable in both X and Y axes directions. In the Z-axis direction, the QDrone improves the tracking behavior, since the tracking error converges to 0 m, as shown in Fig. 4.31. At the same time, the tracking error in the Yaw angle is within ± 0.1 rad.



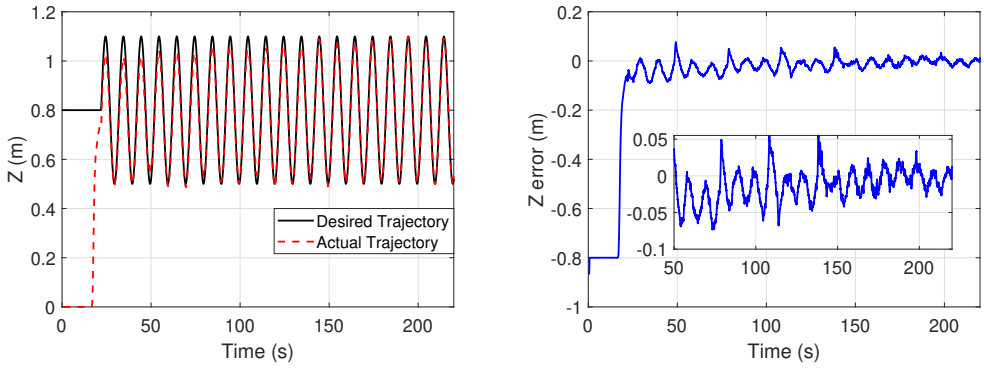
(a) Tracking performance (b) Tracking error

Figure 4.29: Free flight test - translation in the Z-axis direction only : results in the X-axis direction - test 3



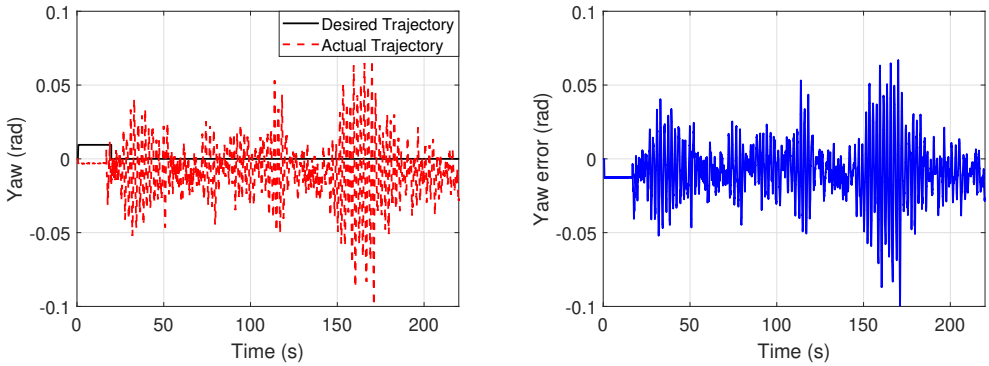
(a) Tracking performance (b) Tracking error

Figure 4.30: Free flight test - translation in the Z-axis direction only : results in the Y-axis direction - test 3



(a) Tracking performance (b) Tracking error

Figure 4.31: Free flight test - translation in the Z-axis direction only : results in the Z-axis direction - test 3



(a) Tracking performance (b) Tracking error

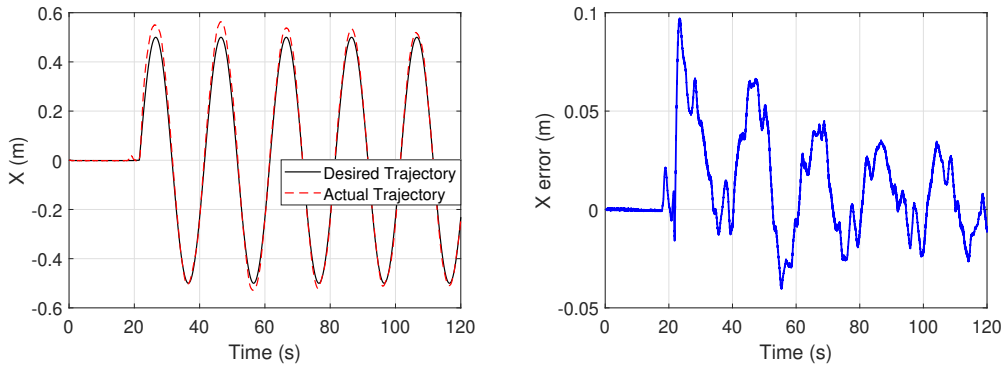
Figure 4.32: Free flight test - translation in the Z-axis direction only : results in the Yaw angle - test 3

Translation in the X+Y-axes Direction

In this test, the QDrone translates in both the X-axis and the Y-axis directions at the same time, and the desired trajectories are the same as the previous individual tests, which are

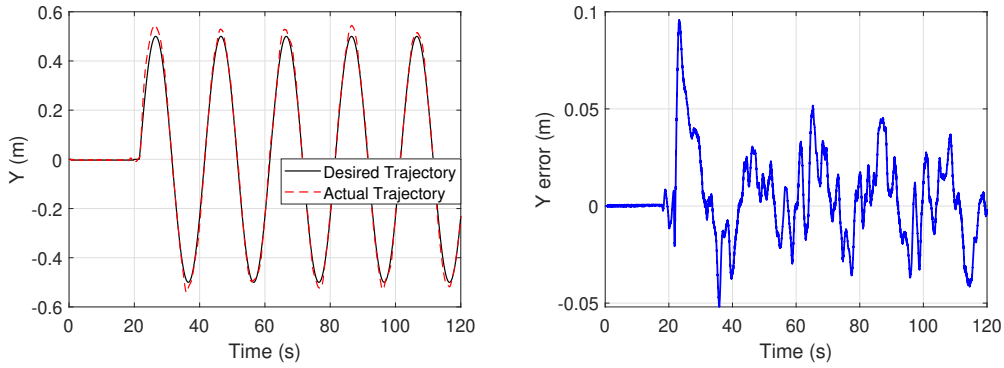
$$\begin{aligned} x_d &= 0.5\sin(\tfrac{\pi}{10}T) \\ y_d &= 0.5\sin(\tfrac{\pi}{10}T) \end{aligned} \tag{4.80}$$

where, $T \in (0, 20]$ is time for each iteration. According to Figs. 4.33 and 4.34, the learning performance in both the X-axis and the Y-axis can be shown, while the tracking error is reduced from around 0.1 m to ± 0.05 m. Also, from Fig. 4.35, the learning result in the Z-axis direction can be seen. Fig. 4.36 shows the flying behavior in the Yaw-axis direction is stable and the tracking error is around ± 0.05 rad.



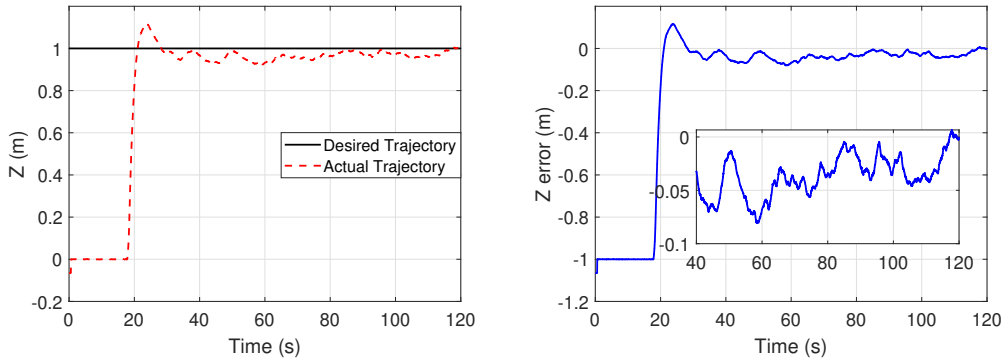
(a) Tracking performance (b) Tracking error

Figure 4.33: Free flight test - translation in the X+Y-axes direction : results in the X-axis direction



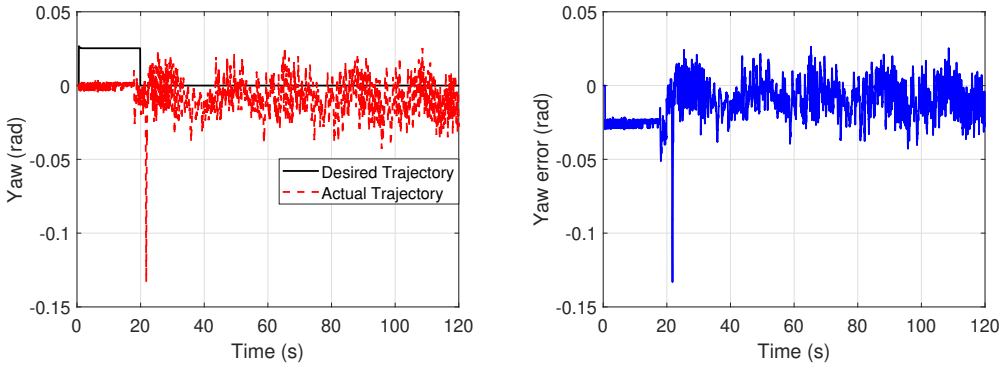
(a) Tracking performance (b) Tracking error

Figure 4.34: Free flight test - translation in the X+Y-axes direction : results in the Y-axis direction



(a) Tracking performance (b) Tracking error

Figure 4.35: Free flight test - translation in the X+Y-axes direction : results in the Z-axis direction



(a) Tracking performance (b) Tracking error

Figure 4.36: Free flight test - translation in the X+Y-axes direction : results in the Yaw angle

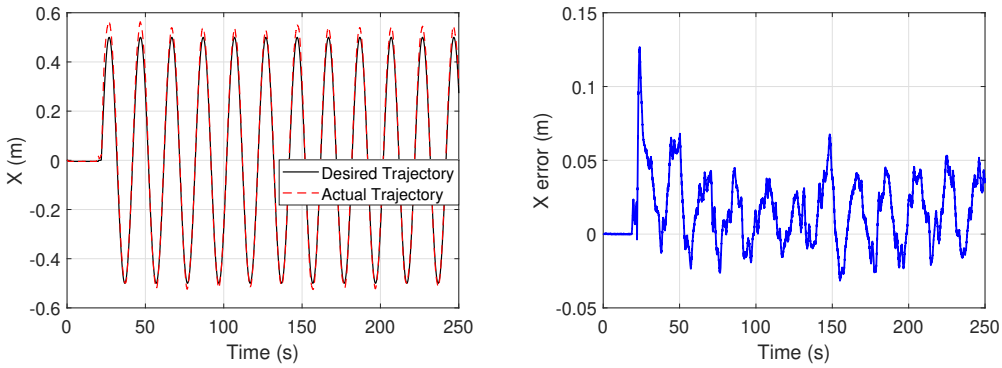
Translation in the X+Y+Z-axes Direction

In this test, the proposed adaptive-type ILC is verified to be working in the three directions at the same time, which are X-axis, Y-axis and Z-axis. The desired trajectories are

$$\begin{aligned} x_d &= 0.5\sin(\tfrac{\pi}{10}T) \\ y_d &= 0.5\sin(\tfrac{\pi}{10}T) \\ z_d &= 1 + 0.3\sin(\tfrac{\pi}{10}T) \end{aligned} \tag{4.81}$$

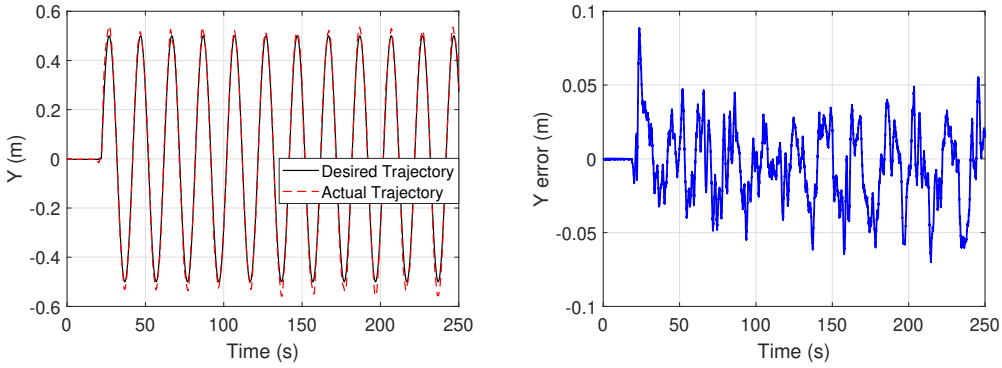
where, $T \in (0, 20]$ is time for each iteration. As shown in Figs. 4.37 and 4.38, the tracking error in the X-axis and the Y-axis directions decreased as iteration time increased and finally are around ± 0.05 m. Meanwhile, the tracking behavior improves in the Z-axis direction, which is shown in Fig. 4.39, since the tracking

error converges to zero. The behavior in the Yaw-axis direction shows the similar performance as the previous test, which is tracking error around ±0.05 rad.



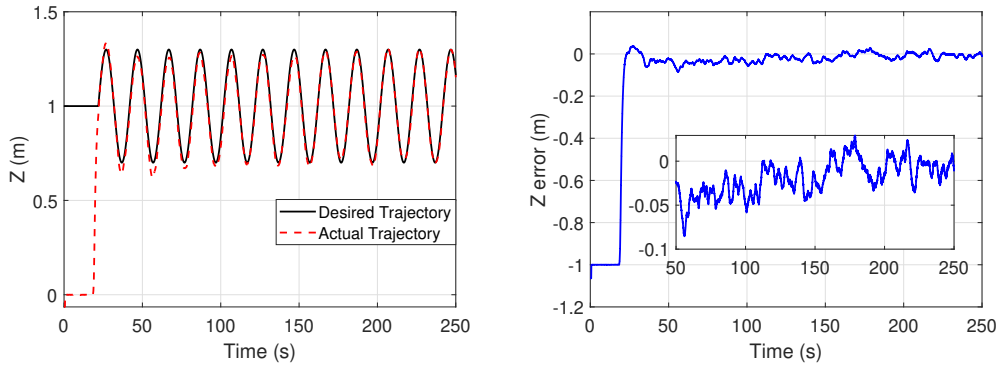
(a) Tracking performance (b) Tracking error

Figure 4.37: Free flight test - translation in the X+Y+Z-axes direction : results in the X-axis direction



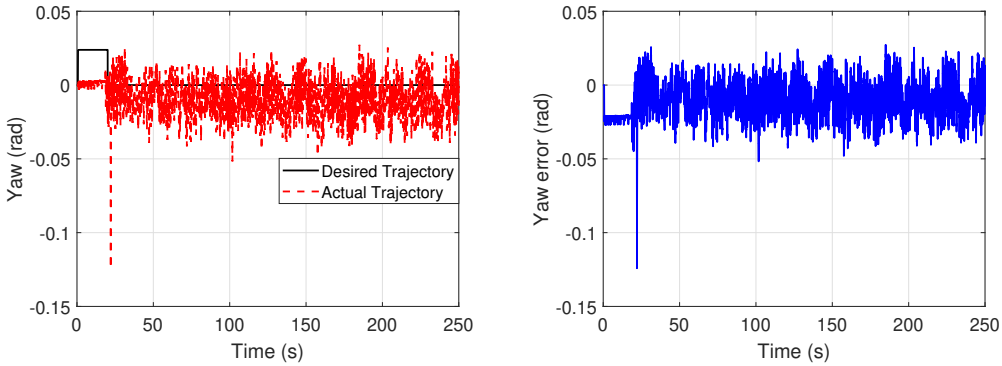
(a) Tracking performance (b) Tracking error

Figure 4.38: Free flight test - translation in the X+Y+Z-axes direction : results in the Y-axis direction



(a) Tracking performance (b) Tracking error

Figure 4.39: Free flight test - translation in the X+Y+Z-axes direction : results in the Z-axis direction



(a) Tracking performance (b) Tracking error

Figure 4.40: Free flight test - translation in the X+Y+Z-axes direction : results in the Yaw angle

4.5 Conclusions

Table 4.1: Conclusion of free flight test - QDrone

Free flight test	Desired trajectory	Tracking error (m)
Translation in the X-axis direction	Eq. (4.75)	$\mid e_{x_i} \mid \leq 0.05$
Translation in the Y-axis direction	Eq. (4.76)	$\mid e_{y_i} \mid \leq 0.05$
Translation in the Z-axis direction test 1	Eq. (4.77)	$\mid e_{z_i} \mid \leq 0.05$
Translation in the Z-axis direction test 2	Eq. (4.78)	$\mid e_{z_i} \mid \leq 0.05$
Translation in the Z-axis direction test 3	Eq. (4.79)	$\mid e_{z_i} \mid \leq 0.05$
Translation in the X+Y-axes direction	Eq. (4.80)	$\mid e_{x_i} \mid \leq 0.05$ $\mid e_{y_i} \mid \leq 0.05$
Translation in the X+Y+Z-axes direction	Eq. (4.81)	$\mid e_{x_i} \mid \leq 0.05$ $\mid e_{y_i} \mid \leq 0.05$ $\mid e_{z_i} \mid \leq 0.05$

In this chapter, the dynamic model of the UAV is presented, and the adaptive-type iterative learning controller is designed based on the attitude and the position separately. The simulation results suggested the possibility of conducting the experiments, and the experimental tests are divided into two parts. First, the test on the Gimbal to verity the proposed attitude controller on the QDrone. According to the results, the proposed controller, successfully drove the QDrone to track

the desired trajectory in the roll and yaw directions, and the learning performance was shown. Second, the tests were run in the flying space, where the QDrone can take-off, then perform the given task. Initially, the QDrone took-off, then the desired trajectory in the X-axis, Y-axis and Z-axis were given separately to verify that each controller matches the desired performance. There were three tests in the Z-axis direction to show the proposed controller fits for different types of desired trajectories. Test 1 verified the desired trajectory can be discontinuous, so that the QDrone can hover for 10s after executing one tracking mission. Test 2 and 3 illustrated the iteration time can be different for the desired trajectory. After that combine trajectories from different directions were considered to test the whole performance. The experimental results clearly demonstrated the proposed adaptive-type iterative learning controller ensure that the QDrone can track all the trajectories, including attitude, position, simple trajectory and complex trajectory. And Table 4.1 shows the tracking error in each test achieves the best accurate of the experimental system, which is the tracking error no larger than 0.05 m.

5 Conclusions and Future Work

This chapter simply summarized and generalized concepts, research, simulation and experimental results. This chapter is organized as beginning with general review of this thesis in Section 5.1, followed by the discussion about future work, which could possibly be further investigated, in Section 5.2.

5.1 General Review

This thesis aims to present two different types of iterative learning control algorithms, and apply them into different areas to verify the capability of them. The first one is PD-type iterative learning controller, which combines the classical PD control approach and the iterative learning technique. The second one is adaptive-type iterative learning controller, which is suitable for the dynamic model with unknown parameters. The general concepts of ILC are included in Chapter 2.

By considering the one rigid manipulator, both controllers are used for simulation

and experimental test, which are described in Chapter 3. The given results shows PD-type iterative learning controller has less parameters to tune and more convenient to employ, however, the convergence speed is less then the adaptive-type one, and there is no response in the first iteration that may cause misunderstanding or risk in real application for factory. Adaptive-type iterative learning controller achieves the objective by using less iteration time, but the degree of difficulty also increased. An adaptive-type iterative learning controller also be used in flexible manipulator with considering to deal with dead-zone problem. An dead-zone inverse was added in the controller to eliminated the effect of dead-zone. The provided experimental results verify the proposed controller remove the dead-zone and other repetitive disturbances successfully.

According to the conclusion of previous chapter, this research was focus on applying adaptive-type iterative learning controller on UAV. The general dynamic model of UAV, control system design, simulation and experimental results are concluded in Chapter 4. The dynamic model could be two parts, one is the attitude, another one is the position. This research uses the traditional inner loop and out loop method to control the UAV. In this case, the control system is designed in two parts, the attitude one and the position one. To verify the performance of the proposed controller, simulation are finished by using Matlab & Simulink and the

experimental tests are done by using the Gimbal and the QDrone. First, mount the QDrone on the Gimbal to test the attitude control involving roll and yaw directions, because there are two degree of freedom when using the Gimabl to do the test. The attached results clearly show the learning process in both roll and yaw directions. Then, the test were run after the QDrone took-off, the test includes X-axis, Y-axis, Z-axis, X+Y-axis and X+Y+Z-axis, all the mentioned test results illustrate the learning performance when the QDrone executed the given missions, and the final results are all achieved the expected requirement which are within 0.05 m.

5.2 Future Work

The work finished in this thesis is preliminary step on the applications of iterative learning control algorithms, which illustrates the proposed two types of iterative learning controller have the capability on dealing with repetitive disturbance both on robotic manipulators and UAVs. However, there are still some issues for the future development.

This research tested one manipulator without any attachments or more links. If there are some attachments on the flexible manipulator, the vibration problem need to be considered [76, 77]. Also, the dynamic model could be more complex, if there are more than one link, and the system becomes the under-actuated one, but the

degrees of freedom increased which makes the manipulator to be agile [78, 79]. This research did not consider the actuator fault, which means the actuator can not provide the desired input, that causes the control performance can not achieve the excepted requirements. Fault tolerant can be an valuable part to be considered in designing the iterative learning algorithm, since fault tolerant control is capable of maintaining the performance of the closed-loop system at an acceptable level in the presence of faults. The actuator fault has been solved in many research [80, 81, 82, 83, 84, 85], but there is a challenge to combine the iterative learning algorithm and the fault tolerant. Therefore, flexible manipulator with attachments, more links dynamic system and actuator fault can be the future work in designing the new iterative learning controllers.

This research still used the traditional control method for the UAV, which is inner loop (attitude control) and out loop (position control). This method has the advantage to tune the gains expediently for each loop. However, the control of the UAV can be done in one loop based on the dynamic model, which means control the attitude and the position at the same time, but the requirement of tuning the gains and the desired trajectory is more strict. This can be a challenge work for the future, to design an iterative learning controller based on differential flatness [86, 87], and how to balance the weights of gains on attitude and position part

can be a research direction in the future work. Meanwhile, quadrotor-manipulator system control can be an interesting topic [88, 89, 90], since it offers a feasible and attractive possibility to pick up and transport desired objects from inaccessible locations where the access of ground vehicles are not possible [91].

Bibliography

[1] Website, https://upload.wikimedia.org/wikipedia/commons/thumb/6/61/Canadarm_1_-_STS-72.jpg/680px-Canadarm_1_-_STS-72.jpg.

[2] Website, https://www.fluxtrends.com/trendconfirmation-food-drone-delivery/.

[3] J. Apkarian, M. Levis, and H. Gurocak, *Rotary Servo Base Unit - Workbook (Student)*, 2011.

[4] B. Wu, X. Cao, and L. Xing, "Robust adaptive control for attitude tracking of spacecraft with unknown dead-zone," *Aerospace Science and Technology*, vol. 45, pp. 196–202, 2015.

[5] J. Iqbal, R. U. Islam, and H. Khan, "Modeling and analysis of a 6 dof robotic arm manipulator," *Canadian Journal on Electrical and Electronics Engineering*, vol. 3, no. 6, pp. 300–306, 2012.

[6] H. A. Almurib, H. F. Al-Qrimli, and N. Kumar, "A review of application industrial robotic design," in *2011 Ninth International Conference on ICT and Knowledge Engineering*. IEEE, 2012, pp. 105–112.

[7] R. S. Penning, J. Jung, J. A. Borgstadt, N. J. Ferrier, and M. R. Zinn, "Towards closed loop control of a continuum robotic manipulator for medical applications," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4822–4827.

[8] B. A. Aikenhead, R. G. Daniell, and F. M. Davis, "Canadarm and the space shuttle," *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, vol. 1, no. 2, pp. 126–132, 1983.

[9] Z. Chen, F. Huang, C. Yang, and B. Yao, "Adaptive fuzzy backstepping control

for stable nonlinear bilateral teleoperation manipulators with enhanced transparency performance," *IEEE transactions on industrial electronics*, vol. 67, no. 1, pp. 746–756, 2019.

[10] A. Brahmi, M. Saad, G. Gauthier, W.-H. Zhu, and J. Ghommam, "Adaptive control of multiple mobile manipulators transporting a rigid object," *International Journal of Control, Automation and Systems*, vol. 15, no. 4, pp. 1779–1789, 2017.

[11] M. A. Wardeh and S. Frimpong, "Kinematic analysis of an under-actuated, closed-loop front-end assembly of a dragline manipulator," *International Journal of Automation and Computing*, pp. 1–12, 2020.

[12] K. S. Eom, I. H. Suh, W. K. Chung, and S.-R. Oh, "Disturbance observer based force control of robot manipulator without force sensor," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 4. IEEE, 1998, pp. 3012–3017.

[13] Z. Zhao, X. He, and C. K. Ahn, "Boundary disturbance observer-based control of a vibrating single-link flexible manipulator," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[14] J. N. Yun and J.-B. Su, "Design of a disturbance observer for a two-link manipulator with flexible joints," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 2, pp. 809–815, 2013.

[15] K.-Y. Chen, "Robust optimal adaptive sliding mode control with the disturbance observer for a manipulator robot system," *International Journal of Control, Automation and Systems*, vol. 16, no. 4, pp. 1701–1715, 2018.

[16] L. Zhang, Q. Jia, G. Chen, and H. Sun, "Pre-impact trajectory planning for minimizing base attitude disturbance in space manipulator systems for a capture task," *Chinese Journal of Aeronautics*, vol. 28, no. 4, pp. 1199–1208, 2015.

[17] W. He, Y. Ouyang, and J. Hong, "Vibration control of a flexible robotic manipulator in the presence of input deadzone," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 48–59, 2017.

[18] W. He, A. O. David, Z. Yin, and C. Sun, "Neural network control of a robotic manipulator with input deadzone and output constraint," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 6, pp. 759–770, 2016.

[19] S. I. Han and J. Lee, "Finite-time sliding surface constrained control for a robot manipulator with an unknown deadzone and disturbance," *ISA transactions*, vol. 65, pp. 307–318, 2016.

[20] W. He, B. Huang, Y. Dong, Z. Li, and C.-Y. Su, "Adaptive neural network control for robotic manipulators with unknown deadzone," *IEEE transactions on cybernetics*, vol. 48, no. 9, pp. 2670–2682, 2017.

[21] Z. Liu, F. Wang, and Y. Zhang, "Adaptive visual tracking control for manipulator with actuator fuzzy dead-zone constraint and unmodeled dynamic," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 10, pp. 1301–1312, 2015.

[22] H. Wang and S. Kang, "Adaptive neural command filtered tracking control for flexible robotic manipulator with input dead-zone," *IEEE Access*, vol. 7, pp. 22 675–22 683, 2019.

[23] C. Hua, L. Zhang, and X. Guan, "Distributed adaptive neural network output tracking of leader-following high-order stochastic nonlinear multiagent systems with unknown dead-zone input," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 177–185, 2015.

[24] S. Khoo, L. Xie, and Z. Man, "Robust finite-time consensus tracking algorithm for multirobot systems," *IEEE/ASME transactions on mechatronics*, vol. 14, no. 2, pp. 219–228, 2009.

[25] Website, https://www.imedicalapps.com/wp-content/uploads/2014/10/iStock_000010884338Large1-e1413268096498.jpg.

[26] Website, https://japanese.alibaba.com/product-detail/danbach-rm08-industrial-robot-manipulator-price-60697719238.html.

[27] S. Sun, "A new method for monitoring machinery movement using an unmanned aerial vehicle (uav) system," Master's thesis, University of Twente, 2019.

[28] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, 2018.

[29] T. Nisser and C. Westin, "Human factors challenges in unmanned aerial vehicles (uavs): A literature review," *School of Aviation of the Lund University, Ljungbyhed*, 2006.

[30] J. Everaerts *et al.*, "The use of unmanned aerial vehicles (uavs) for remote sensing and mapping," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, no. 2008, pp. 1187–1192, 2008.

[31] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018.

[32] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *Ieee Access*, vol. 7, pp. 48 572–48 634, 2019.

[33] E. Cheng, *Aerial photography and videography using drones*. Peachpit Press, 2015.

[34] B. Shirani, M. Najafi, and I. Izadi, "Cooperative load transportation using multiple uavs," *Aerospace Science and Technology*, vol. 84, pp. 158–169, 2019.

[35] A. S. Aghdam, M. B. Menhaj, F. Barazandeh, and F. Abdollahi, "Cooperative load transport with movable load center of mass using multiple quadrotor uavs," in *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*. IEEE, 2016, pp. 23–27.

[36] D. Giordan, M. S. Adams, I. Aicardi, M. Alicandro, P. Allasia, M. Baldo, P. De Berardinis, D. Dominici, D. Godone, P. Hobbs *et al.*, "The use of unmanned aerial vehicles (uavs) for engineering geology applications," *Bulletin of Engineering Geology and the Environment*, pp. 1–45, 2020.

[37] E. Romero-Chambi, S. Villarroel-Quezada, E. Atencio, M.-L. Rivera *et al.*,

"Analysis of optimal flight parameters of unmanned aerial vehicles (uavs) for detecting potholes in pavements," *Applied Sciences*, vol. 10, no. 12, p. 4157, 2020.

[38] V. Spurnỳ, T. Báča, M. Saska, R. Pěnička, T. Krajník, J. Thomas, D. Thakur, G. Loianno, and V. Kumar, "Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles," *Journal of Field Robotics*, vol. 36, no. 1, pp. 125–148, 2019.

[39] K. Kuru, D. Ansell, W. Khan, and H. Yetgin, "Analysis and optimization of unmanned aerial vehicle swarms in logistics: An intelligent delivery platform," *Ieee Access*, vol. 7, pp. 15 804–15 831, 2019.

[40] S. Goudarzi, N. Kama, M. H. Anisi, S. Zeadally, and S. Mumtaz, "Data collection using unmanned aerial vehicles for internet of things platforms," *Computers & Electrical Engineering*, vol. 75, pp. 1–15, 2019.

[41] R. Pěnička, J. Faigl, M. Saska, and P. Váňa, "Data collection planning with non-zero sensing distance for a budget and curvature constrained unmanned aerial vehicle," *Autonomous Robots*, vol. 43, no. 8, pp. 1937–1956, 2019.

[42] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[43] T. Chen and J. Shan, "A novel cable-suspended quadrotor transportation system: From theory to experiment," *Aerospace Science and Technology*, vol. 104, p. 105974, 2020.

[44] A. Hock and A. P. Schoellig, "Distributed iterative learning control for a team of quadrotors," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 4640–4646.

[45] K. L. Moore, "Iterative learning control: An expository overview," in *Applied and computational control, signals, and circuits*. Springer, 1999, pp. 151–214.

[46] X. Li, Q. Ren, and J.-X. Xu, "Precise speed tracking control of a robotic fish via iterative learning control," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2221–2228, 2015.

[47] P. Bondi, G. Casalino, and L. Gambardella, "On the iterative learning control theory for robotic manipulators," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 14–22, 1988.

[48] Y. Yang, Z. Liu, and G. Ma, "Adaptive distributed control of a flexible manipulator using an iterative learning scheme," *IEEE Access*, vol. 7, pp. 145 934–145 943, 2019.

[49] M. Zhu, L. Ye, and X. Ma, "Estimation-based quadratic iterative learning control for trajectory tracking of robotic manipulator with uncertain parameters," *IEEE Access*, vol. 8, pp. 43 122–43 133, 2020.

[50] R. Adlakha and M. Zheng, "An optimization-based iterative learning control design method for uav's trajectory tracking," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1353–1359.

[51] M. Zhaowei, H. Tianjiang, S. Lincheng, K. Weiwei, Z. Boxin, and Y. Kaidi, "An iterative learning controller for quadrotor uav path following at a constant altitude," in *2015 34th Chinese Control Conference (CCC)*. IEEE, 2015, pp. 4406–4411.

[52] W. Giernacki, "Optimal tuning of altitude controller parameters of unmanned aerial vehicle using iterative learning approach," in *Conference on Automation*. Springer, 2019, pp. 398–407.

[53] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE control systems magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[54] Z. Bien and J.-X. Xu, *Iterative learning control: analysis, design, integration and applications*. Springer Science & Business Media, 2012.

[55] S. Arimoto, "Iterative learning control for robot systems," *Proc. IECON, Tokyo, 1984*, pp. 393–398, 1984.

[56] J.-X. Xu, "A survey on iterative learning control for nonlinear systems," *International Journal of Control*, vol. 84, no. 7, pp. 1275–1294, 2011.

[57] J.-X. Xu and Y. Tan, *Linear and nonlinear iterative learning control*. Springer, 2003, vol. 291.

[58] C. Chen, C. Wen, Z. Liu, K. Xie, Y. Zhang, and C. P. Chen, "Adaptive asymptotic control of multivariable systems based on a one-parameter estimation approach," *Automatica*, vol. 83, pp. 124–132, 2017.

[59] C. Chen, Z. Liu, K. Xie, Y. Zhang, and C. P. Chen, "Asymptotic adaptive control of nonlinear systems with elimination of overparametrization in a nussbaum-like design," *Automatica*, vol. 98, pp. 277–284, 2018.

[60] C. Chen, C. Wen, Z. Liu, K. Xie, Y. Zhang, and C. P. Chen, "Adaptive consensus of nonlinear multi-agent systems with non-identical partially unknown control directions and bounded modelling errors," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4654–4659, 2016.

[61] S. Mobayen, F. Tchier, and L. Ragoub, "Design of an adaptive tracker for n-link rigid robotic manipulators based on super-twisting global nonlinear sliding mode control," *International Journal of Systems Science*, vol. 48, no. 9, pp. 1990–2002, 2017.

[62] T. Sun, H. Pei, Y. Pan, H. Zhou, and C. Zhang, "Neural network-based sliding mode adaptive control for robot manipulators," *Neurocomputing*, vol. 74, no. 14-15, pp. 2377–2384, 2011.

[63] W. Rudin, "Principles of mathematical analysis, new york, mcgtaw-hill," 1976.

[64] B. Park, T.-Y. Kuc, and J. S. Lee, "Adaptive learning control of uncertain robotic systems," *International Journal of Control*, vol. 65, no. 5, pp. 725–744, 1996.

[65] T. Chen and J. Shan, "Distributed control of multiple flexible manipulators with unknown disturbances and dead-zone input," *IEEE Transactions on Industrial Electronics*, 2019.

[66] L. Gümüsel and N. G. Özmen, "Modelling and control of manipulators with flexible links working on land and underwater environments," *Robotica*, vol. 29, no. 3, pp. 461–470, 2011.

[67] Z. Liu, J. Liu, and W. He, "An adaptive iterative learning algorithm for boundary control of a flexible manipulator," *International Journal of Adaptive Control and Signal Processing*, vol. 31, no. 6, pp. 903–916, 2017.

[68] T. Meng and W. He, "Iterative learning control of a robotic arm experiment platform with input constraint," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 1, pp. 664–672, 2017.

[69] S. Yang, J.-X. Xu, X. Li, and D. Shen, *Iterative learning control for multi-agent systems coordination*. John Wiley & Sons, 2017.

[70] J. Apkarian, M. Levis, and H. Gurocak, *SRV02 Base Unit User Manual*, 2011.

[71] T. Chen, M. Li, and J. Shan, "Iterative learning control of a flexible manipulator considering uncertain parameters and unknown repetitive disturbance," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 2209–2214.

[72] P.-i. Pipatpaibul and P. Ouyang, "Quadrotor uav control: online learning approach," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 54839, 2011, pp. 701–710.

[73] Y.-j. Zhong, Z.-x. Liu, Y.-m. Zhang, W. Zhang, and J.-y. Zuo, "Active fault-tolerant tracking control of a quadrotor with model uncertainties and actuator faults," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 1, pp. 95–106, 2019.

[74] Z. Ma and G. Sun, "Dual terminal sliding mode control design for rigid robotic manipulator," *Journal of the Franklin Institute*, vol. 355, no. 18, pp. 9127–9149, 2018.

[75] *Research Studio Setup Guide*, 2018.

[76] Z. Liu and J. Liu, "Dynamic modeling and vibration control for a nonlinear three-dimensional flexible manipulator," in *PDE Modeling and Boundary Control for Flexible Mechanical System*. Springer, 2020, pp. 137–171.

[77] Z. Mohamed, J. Martins, M. Tokhi, J. S. Da Costa, and M. Botto, "Vibration control of a very flexible manipulator system," *Control Engineering Practice*, vol. 13, no. 3, pp. 267–277, 2005.

[78] W. He, H. Gao, C. Zhou, C. Yang, and Z. Li, "Reinforcement learning con-

trol of a flexible two-link manipulator: an experimental investigation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[79] B. Subudhi and A. S. Morris, "Dynamic modelling, simulation and control of a manipulator with flexible links and joints," *Robotics and Autonomous Systems*, vol. 41, no. 4, pp. 257–270, 2002.

[80] Y. Wang, J. Shi, D. Zhou, and F. Gao, "Iterative learning fault-tolerant control for batch processes," *Industrial engineering chemistry research*, vol. 45, no. 26, pp. 9050–9060, 2006.

[81] Z. Yu, Y. Qu, and Y. Zhang, "Safe control of trailing uav in close formation flight against actuator fault and wake vortex effect," *Aerospace Science and Technology*, vol. 77, pp. 189–205, 2018.

[82] Y. Zhong, Y. Zhang, W. Zhang, J. Zuo, and H. Zhan, "Robust actuator fault detection and diagnosis for a quadrotor uav with external disturbances," *IEEE Access*, vol. 6, pp. 48 169–48 180, 2018.

[83] Y. Zhong, Y. Zhang, and W. Zhang, "Active fault-tolerant tracking control of a quadrotor uav," in *2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*. IEEE, 2018, pp. 497–502.

[84] Z. Liu, C. Yuan, Y. Zhang, and J. Luo, "A learning-based fault tolerant tracking control of an unmanned quadrotor helicopter," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 145–162, 2016.

[85] Z. Liu, C. Yuan, and Y. Zhang, "Active fault-tolerant control of unmanned quadrotor helicopter using linear parameter varying technique," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 415–436, 2017.

[86] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[87] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6567–6572.

[88] H. Yang and D. Lee, "Dynamics and control of quadrotor with robotic manipulator," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5544–5549.

[89] Y. Qi, J. Wang, and J. Shan, "Aerial cooperative transporting and assembling control using multiple quadrotor–manipulator systems," *International Journal of Systems Science*, vol. 49, no. 3, pp. 662–676, 2018.

[90] Z. A. Ali and X. Li, "Controlling of an under-actuated quadrotor uav equipped with a manipulator," *IEEE Access*, vol. 8, pp. 34 664–34 674, 2020.

[91] T. Wang, K. Umemoto, T. Endo, and F. Matsuno, "Dynamic hybrid position/force control for the quadrotor with a multi-degree-of-freedom manipulator," *Artificial Life and Robotics*, vol. 24, no. 3, pp. 378–389, 2019.