HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

—o0o—

Data-driven Optimal Tracking Control for a Quadrotor

PHAM DINH DUONG
NGUYEN TAT CHUNG

Control Engineering and Automation
Automatic Control

Advisor: **Assoc.Prof. Dao Phuong Nam**

Hanoi, 06/2021

Abstract

Over the past decades, optimal control has been extensively researched and applied in the field of control and automation. Solving the optimal control problem is often reduced to solving the Halmilton-Jacobi-Bellman (HJB) equation. However, this is a rather complicated nonlinear differential equation with no analytical solution. The idea that has been proposed and received most of the research attention is methods for approximating solutions to the HJB equation. In parallel with that, with the development of Reinforcement Learning algorithms, the birth of Adaptive Dynamic Programming with the approximation of the solution of the HJB equation using Neural Networks (NNs) with a Actor-Critic structure has opened up many development directions: On-policy with Online Actor-Critic, Online IRL,... Off-policy with Off-policy IRL, Data-Driven Policy Interation (PI),... Each method has its own advantages and disadvantages. With On-policy, although information is required for all or part of the system, the algorithm is adjusted directly during the control process. With Off-policy, the advantage is that it does not require complete dynamic information of the system, but it is necessary to collect and process data before applying control. Within the framework of the project, we focus on Off-policy (Data-Driven) algorithms based on PI to find optimal controller for linear and nonlinear systems with unknown models. Furthermore, while the majority of research addresses the problem of asymptotically stable optimization for the system, our project is interested in the problem of optimal tracking control, thereby introducing an additional component of the decay factor for the system's cost function. The algorithm is developed generally for the Affine system, with examples presented in each section for both linear and nonlinear systems, accompanied by simulations to test the effectiveness of the algorithm.

To show the applicability of the algorithm, we apply the above control scheme for a 4-wing aircraft (Quadrotor). In recent years, drones have received great attention in the research community due to their enormous application potential. As a typical type of unmanned aircraft, quadrotors have become increasingly popular thanks to their ability to take off and land vertically, position stability and flexible trajectory. Many controller design methods have been proposed for quadrotors: PID controller, backstepping, sliding mode

control,... However, in reality, the quadrotor is a nonlinear system with 6 degrees of freedom and high coupling, the exact model of the system is difficult to obtain (due to the complex structure, because when operating, additional equipments is attached or carrying or pulling other unknown objects), designing a controller structure that does not need the model but still ensures quality control is essential. Based on that, we investigated and applied the Data-driven PI algorithm to solve the Optimal tracking control problem for quadrotors. Simulations are conducted on MATLAB. Finally, there are comments and directions for future algorithm development.

Hanoi, 16-06-2021
Students

Pham Dinh Duong

Nguyen Tat Chung

Abbreviations

$\|.\|$	Distance in Euclidean space.
RL	Reinforcement Learning.
IRL	Integral Reinforcement Learning.
HJB	Hamilton–Jacobi–Bellman.
PI	Policy Iteration.
ADP	Approximate/Adaptive Dynamic Programming.
NN	Neural Network.
CNN	Critic Neural Network.
ANN	Actor Neural Network.
AC	Actor-Critic.
PE	Persistent Excitation Condition.

List of figures

Table of contents

Chapter 1

Overview of Reinforcement Learning and Adaptive Dynamic Programming

1.1 Overview of Reinforcement Learning(RL)

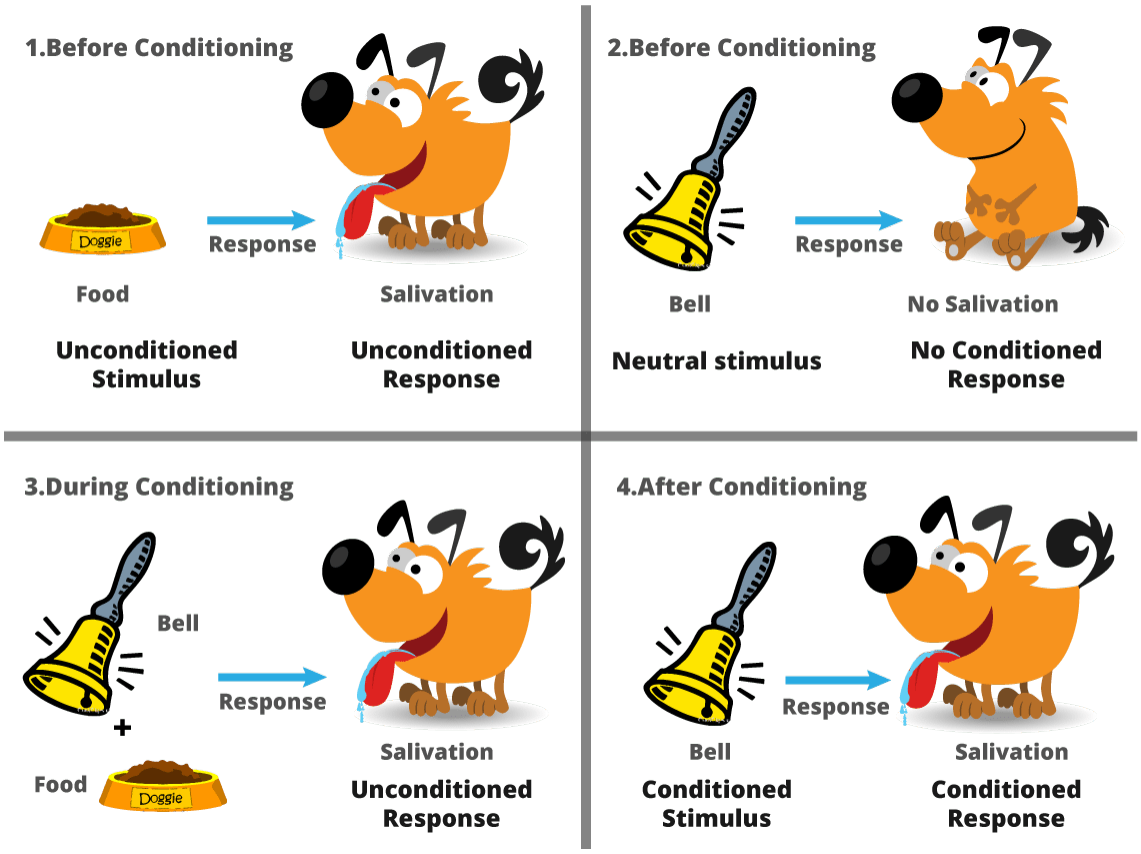


Figure 1.1: An example of RL in animals

Every living organism interacts with the environment and uses that interaction to regulate behavior in order to survive and develop. We call this behavioral adjustment based on interaction with the environment: Reinforcement Learning (RL). RL involves an object (agent) interacting with its environment, modifying its actions (also known as a "control policy") based on

feedback from previous actions. This adjustment is made using assessable information from the environment. In other words, RL is based on the cause-effect relationship of behavior and reward (or punishment). RL algorithms start from the idea that successful behaviors (which yield high rewards) will be remembered, in the sense that they tend to be reused on subsequent occasions [9]. The idea of RL originates from experiments on biological learning. RL is theoretically closely connected directly and indirectly with adaptive optimal control methods.

One of the popular structures of RL is the Actor-Critic structure [Barto, Sutton, Anderson 1983], in which an Actor component performs actions (control policies) that affect the environment, and a The Critic section evaluates that action. Based on that assessment, many methods are used to calibrate or improve the action so that the new action will create better value (reward) than the previous value. In summary, the Actor-Critic structure consists of two steps: evaluating behavior and improving behavior. Behavioral assessment is done by observing the results returned from the environment after performing the behavior.

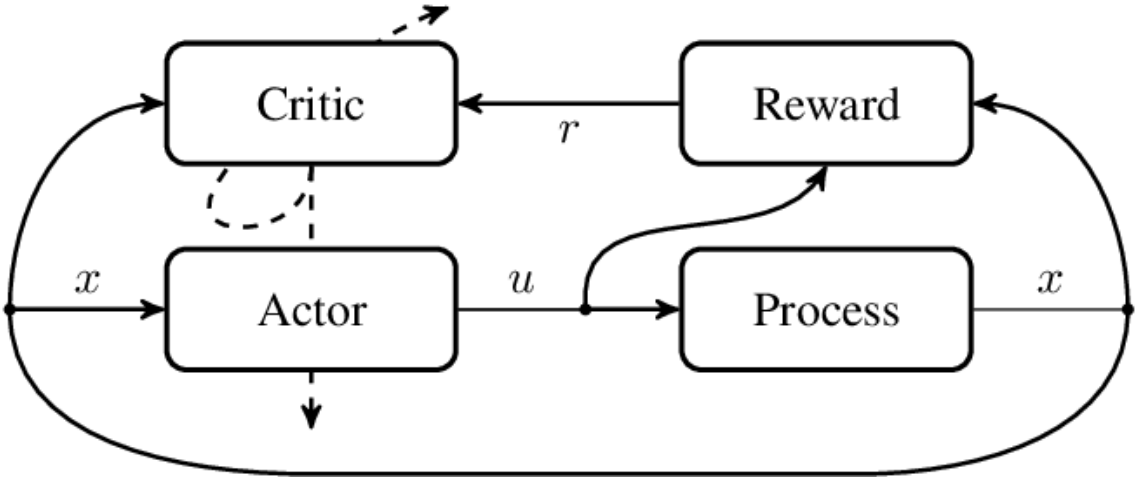


Figure 1.2: Actor-Critic structure in Reinforcement Learning

Returning to learning from nature, survival in nature is both extremely important and extremely difficult. Therefore, individuals often have to try to perform behaviors that both retain maximum energy and achieve maximum behavioral goals. Such behavior is called optimal behavior. Optimal behavior is often based on the following criteria: minimizing energy costs, risks,... maximizing rewards,... Therefore, research on RL with Actor-Critic structure in which Critic evaluates behavior based on optimal standards is natural and necessary.

Feedback control is the study of methods for designing and developing controllers based on system feedback, in order to achieve desired quality of control and safety. These systems include: aircraft, ships, automobiles, robotic

systems, industrial processes, temperature control systems, weather systems, and many other objects. Imitating and learning from nature to design optimal control (optimization is often understood as achieving the control goal by consuming the least amount of resources) is well-founded and necessary. Next, the project would like to briefly present the combination of RL and feedback control: adaptive dynamic programming algorithm.

1.2 Overview of Adaptive Dynamic Programming (ADP)

1.2.1 Bellman's principles of dynamic programming

Mathematician Richard Bellman invented the "Dynamic Programming" method in 1957. R.Bellman's optimization principle is stated as follows [1]:
"Optimize n steps by optimizing all paths to step $n-1$ and choosing the path with total cost from step 1 to step $n-1$ and from $n-1$ up to n is the lowest (highest reward)".

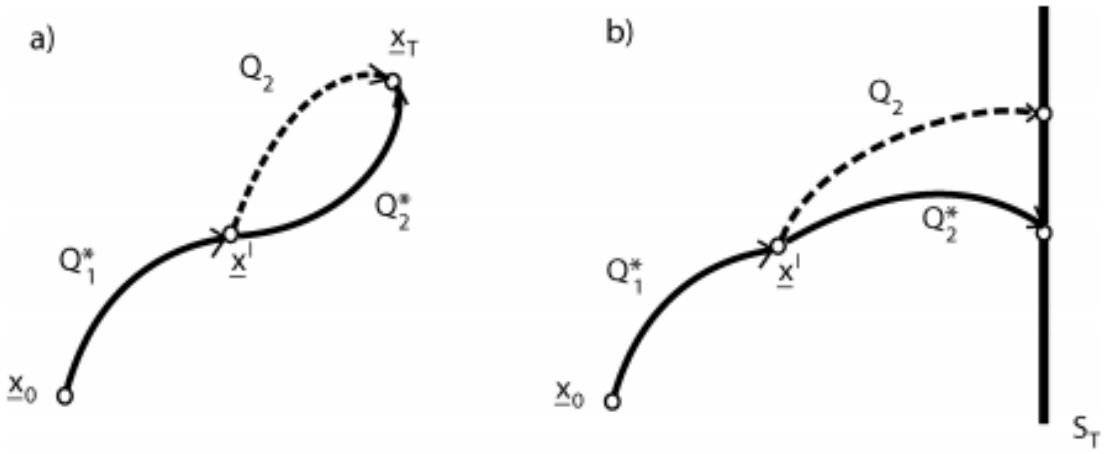


Figure 1.3: Bellman's principles [16]

1.2.2 HJB equation and adaptive dynamic programming

Consider the affine dynamic system:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{1.1}$$

Where $x \in \mathbb{R}^n$ is the states vector, $u =\in \mathbb{R}^m$ is the control input, $f(x(t)) \in \mathbb{R}^n$ and $g(x(t)) \in \mathbb{R}^{n\times m}$ Suppose $f(0) = 0$ và $f(x(t)) + g(x(t))u(x(t))$ satisfies

Lipschitz condition continously in a region $\Omega \subseteq \mathbb{R}^n$. The cost function:

$$V(x) = \int_t^\infty r(x,u)d\tau \tag{1.2}$$

Where $r(x,u) = Q(x) + u^T Ru$. $Q(x)$ is a positive-definite function, R is a positive semi-definite matrix. Based on Optimal control theory [16], the optimal control signal to stabilize the system and minimize the above cost function is:

$$u^*(x) = -\frac{1}{2}R^{-1}g^T(x)\frac{\partial V^*(x)^T}{\partial x} \tag{1.3}$$

With $V^*(\underline{x},t) = \min_{\underline{u}\in\mathbb{U}} \int_t^T r(\underline{x},\underline{u},\tau)d\tau$ Therefore, we have the HJB equation for a affine system:

$$\begin{cases} H^*(\underline{x},\underline{u}^*,V_{\underline{x}}^*) = \frac{\partial V^*(\underline{x})}{\partial \underline{x}}\left(\underline{f}(\underline{x}) + \underline{g}(\underline{x})\underline{u}^*\right) + Q(\underline{x}) + \underline{u}^{*T}R\underline{u}^* = 0 \\ V^*(\underline{0}) = 0 \end{cases} \tag{1.4}$$

If the system is linear and the penalty function is quadratic with states and control signals, the optimal controller can be written as a state feedback controller, with the Gain matrix as the solution to the Riccati equation. However, if the system is nonlinear, we must solve the HJB equation to find the optimal solution. However, solving the HJB equation is really difficult because of the nonlinearity of this partial nonlinear equation. The more nonlinear and complex the system is, the more difficult the equation is to solve.
In the early times, people used Dynamic Programming to solve optimization problems, typically in 1957 with the introduction of Bellman's principle: An optimal control law has the property that: with arbitrary initial state and arbitrary first decision, subsequent decisions must sum to optimality. This principle can be simply expressed as a regression formula. Therefore, this principle is widely applied, from discrete to continuous systems, from linear to nonlinear systems.
However, using the Dynamic Programming principle to solve the HJB equation for nonlinear systems has many weaknesses. As the dimensionality of state x and control signal u increases, implementing the algorithm becomes more difficult, even impossible to solve due to the requirement of an extremely large amount of backward calculation ("curse of dimensionality" problem). To address these weaknesses, Werbos proposed the Adaptive Dynamic Programing architecture, in which the main idea is to use function approximation (for example, Neural networks, Fuzzy model, Polynomial,...) to approximate the cost function and solve the dynamic programming problem forward in time.

The ADP algorithm was developed, using function approximation to approximate the cost function (Critic) and the control law (Actor), this famous

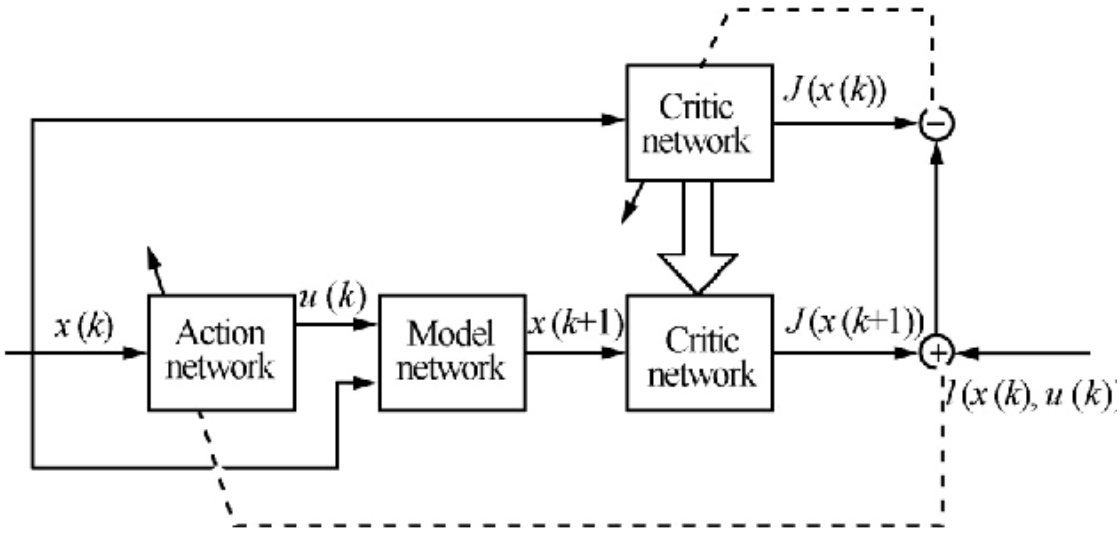


Figure 1.4: Actor-Critic Structure in ADP

structure is briefly called Actor-Critic. Updating the weights for these two function approximations is done in two ways, and also opens up two types of algorithms: On-policy and Off-policy. With On-policy, control laws are updated directly during the control process. Several methods have been developed in recent years, including: Online Actor-Critic updates weights in parallel with knowing the system dynamics information or using a system identifier (Identifier); Online Integral Reinforment Learning (IRL) eliminates the need to know the internal dynamics of the system, but still needs to know the information of $g(x)$. With Off-policy, two control laws are separated: an acceptable control law, with disturbance requirements ensuring the PE condition, is used to generate data of the system's dynamics, then data (mainly state x and control signal u) are trained to find the optimal controller. The strength of the Off-policy algorithm is that we absolutely do not need dynamic information of the system, this is simply explained: thanks to the large amount of previously collected data, essentially the dynamic information of the system has been indirectly expressed in this amount of data. In the field of Optimal control using the ADP algorithm, research focuses mainly on developing algorithms for the problem of system stability control (the problem of bringing the system state asymptotically to the origin 0). Unlike the stability control problem in which the control signal u approaches 0 when the system is stable, the tracking control problem requires holding a control signal u other than 0 when the system is on the desired trajectory. Some studies use the method of separating the controller into 2 parts, one component is the Feed-forward control signal - the signal needed to keep the system on orbit, the other component is the signal to drive the system to a stable state, based on the principle of optimization [7]. However, it is easy to see that this

method is only partially optimal, not completely. A more general treatment is to use a discount factor in the cost function, a common method used in RL in Machine Learning algorithms. From this penalty function, the ADP algorithm is deployed to find the optimal controller for the system. In this project, we focus on researching this treatment method, using Off-policy ADP to solve the Optimal tracking control problem with the cost function modified by adding a decay factor. The remaining layout of the project is shown as follows: Chapter 2 will present the general Data-driven PI algorithm for the affine system. Specifically, section 2.1 introduces the Optimal tracking control problem, section 2.2 presents the Data-driven PI algorithm to solve the problem in 2.1, section 2.3 gives examples of applying the algorithm to linear and nonlinear systems, accompanied by simulation verification. Chapter 3 presents the application of the Data-driven PI algorithm to find the optimal controller for the Quadrotor. Specifically, section 3.1 briefly introduces the Quadrotor, section 3.2 provides the kinematic and dynamic model of the object, section 3.3 outlines the Quadrotor control method in general, including 2 control loops for position and attitude angle, section 3.4 presents the specific Data-driven PI algorithm for 2 control loops of Quadrotor, section 3.5 gives simulation results applying this algorithm. Finally, there are conclusions and directions for future research and project development.

Chapter 2

Data-driven PI algorithm for Optimal tracking control problem (OTCP)

2.1 Optimal Tracking Control Problem

Consider a affine system:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{2.1}$$

Where $x = [x_1...x_n]^T \in \mathbb{R}^n$ is state-system vector, $u = [u_1...u_n] \in \mathbb{R}^m$ is control input, $f(x(t)) \in \mathbb{R}^n$ and $g(x(t)) \in \mathbb{R}^{n\times m}$ Suppose $f(0) = 0$ và $f(x(t)) + g(x(t))u(x(t))$ satisfy the Lipschitz condition continuously in a region $\Omega \subseteq \mathbb{R}^n$. The above affine nonlinear system is able to be stable in the region Ω, meaning that there exists a control signal $u(t)$ for (2.1) to be asymptotically stable in the region Ω. Define tracking errors:

$$e_d(t) = x(t) - x_d(t) \tag{2.2}$$

Where $x_d(t) \in \mathbb{R}^n$ is the desired trajectory. The cost function is defined as follows:

$$V(e_d(t), x_d(t)) = \int_t^\infty e^{-\lambda(\tau-t)}(e_d(\tau)^T Q_e e_d(\tau) + u(\tau)^T R u(\tau))d\tau \tag{2.3}$$

Here λ is the decay factor, $Q_e \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are the positive definite symmetric matrices. Note that the above cost function contains the cost of the tracking error and the entire cost of the control signal acting, to ensure that the system can both track the trajectory and the cost of control is optimal. To control the system (2.1) to optimally follow the trajectory $x_d(t)$, we need to choose the control signal $u(t)$ to minimize the cost function (2.3) for any signal $x(t)$. Here, we assume that the trajectory $x_d(t)$ is satisfied

$$\dot{x}_d(t) = r_d(x_d(t)), \quad r_d(0) = 0 \tag{2.4}$$

Here $x_d(t)$ is bounded and the function $r_d(x_d(t)) \in \mathbb{R}^n$ is a continuous Lipschitz function. In the existing solutions for *Optimal Tracking Control (OTCP)* in [7], the optimal control signal consists of a feedforward component $u_d(t)$ and an optimal feedback control signal $u_e(t)$. Suppose the optimal trajectory $x_d(t)$ is satisfied

$$\dot{x}_d(t) = f(x_d(t)) + g(x_d(t))u_d(t) \tag{2.5}$$

The feedforward control signal $u_d(t)$ can be calculated if the system dynamics $f(x(t))$, $g(x(t))$ are known and $g^{-1}(x)$ exists:

$$u_d(t) = g^{-1}(x_d(t))(\dot{x}_d(t) - f(x_d(t))) \tag{2.6}$$

The optimal response signal $u_e(t)$ is designed to optimize the following cost function:

$$V_e(e_d(t)) = \int_t^\infty (e_d(\tau)^T Q_e e_d(\tau) + u_e(\tau)^T R u_e(\tau))d\tau \tag{2.7}$$

And it can be obtained by solving the HJB equation for the expression (2.7):

$$u_e^* = -\frac{1}{2}R^{-1}g(x(t))^T \frac{\partial V_e(e_d(t))}{\partial e_d} \tag{2.8}$$

Remark 2.1.1. *In standard methods for solving the OTCP problem, the signal $u_d(t)$ requires knowing the dynamical model of the system, both $f(x)$ and $g(x)$. While the control signal $u_e(t)$ needs to at least know $g(x)$.*

2.2 The Data-driven Policy Iteration (PI) algorithm for OTCP with completely unknown model

In this section, we would like to present the Data-driven PI algorithm for the OTCP problem presented above. To do so, we need to measure the state x and the control signal u, while we already know the desired trajectory $x_d(t)$.

2.2.1 The system dynamic equation

In this step, the system dynamic equation is rewritten in an expanded form as follows, this expanded system combines the system kinematic error $e_d(t)$ and the set trajectory $x_d(t)$.

$$\dot{X}(t) = F(X(t)) + G(X(t))u(t) \tag{2.9}$$

Where $X(t) = [e_d(t)^T \ x_d(t)^T]^T \in \mathcal{X} \subset \mathbb{R}^{2n}$ and

$$F(X(t)) = \begin{bmatrix} f(e_d(t) + x_d(t)) - r_d(x(t)) \\ r_d(x_d(t)) \end{bmatrix} \tag{2.10}$$

$$G(X(t)) = \begin{bmatrix} g(e_d(t) + x_d(t))) \\ 0 \end{bmatrix} \tag{2.11}$$

Define the cost function:

$$V(X(t)) = \int_t^\infty e^{-\lambda(\tau - t)}[X(\tau)^T Q X(\tau) + u(\tau)^T R u(\tau)]d\tau \tag{2.12}$$

$$Q = \begin{bmatrix} Q_e & 0 \\ 0 & 0 \end{bmatrix} \tag{2.13}$$

The decay factor λ in the cost function is necessary according to [14] [13] [8], if the desired trajectory does not go to zero as time approaches infinity, the cost function with $\lambda = 0$ will be unbounded because the feedforward component $u_d(t)$ depends on the desired state trajectory. To eliminate that drawback, the $e^{-\lambda(\tau-t)}$ [17] component is added to handle the above cases, for example where the desired orbit is a bounded cyclic signal.

Definition 2.2.1. *A signal $u(X)$ is defined as admissible in the region $\mathcal{X}$, denoted $u(X) \in \mathcal{U}(\mathcal{X})$ if $u(X)$ is continuous on the region $\mathcal{X}$, $u(0) = 0$, $u(X)$ stabilizes the system (2.9) on $\mathcal{X}$ and $V(X)$ are bounded $\forall X \in \mathcal{X}$*

Differentiating $V(X(t))$ in (2.12) with respect to time, we get:

$$\begin{aligned} \dot{V}(X) &= \lambda \int_t^\infty e^{-\lambda(\tau - t)}[X(\tau)^T Q X(\tau) + u(\tau)^T R u(\tau)]d\tau - X^T Q X - u^T R u \\ &= \lambda V(X) - X^T Q X - u^T R u \end{aligned}$$

Based on that, the HJB equation can be rewritten as:

$$H(X, u^*, \nabla V^*(X)) = X^T Q X + u^{*T} R u^* - \lambda V^*(X) + \nabla V^*(X)^T (F(X) + G(X)) u^*) = 0 \tag{2.14}$$

With $\nabla V^*(X) = \partial V^*(X)/\partial X$ and $V^*(X)$ is the optimal cost function:

$$V^*(X(t)) = \min_{u \in \mathcal{U}(\Omega)} \int_t^\infty e^{-\lambda(\tau - t)}[X(\tau)^T Q X(\tau) + u(\tau)^T R u(\tau)]d\tau \tag{2.15}$$

And the optimal control signal:

$$u^*(X) = \underset{u \in \mathcal{U}(\Omega)}{argmin}\, H(X, u, \nabla V^*) = -\frac{1}{2} R^{-1} G(X)^T \nabla V^*(X) \tag{2.16}$$

The optimal control signal can be solved using the standard PI algorithm as follows: 1. Policy Evaluation

$$[\nabla V^{i+1}(X)]^T (F(X) + G(X)u^i) + X^T Q X + [u^i]^T R u^i - \lambda V^{i+1}(X) = 0 \tag{2.17}$$

2. Policy Improvement

$$u^{i+1}(X) = -\frac{1}{2} R^{-1} G(X)^T \nabla V^{i+1}(X) \tag{2.18}$$

Theorem 2.2.2. *$u^i(X) \in \mathcal{U}(\mathcal{X})$ and $V^{i+1}(X) \in \mathcal{V}(\mathcal{X})$ satisfy (2.17) with the condition $V^{i+1}(0) = 0$. Then, the control signal (2.18) is amissible in $\mathcal{X}$, $\forall i \geq 1$. Furthermore, if $V^{i+1}(X)$ is the only positive definite function satisfying (2.17) with the condition $V^{i+1}(0) = 0$ then $V^*(X) \leq V^{i+1}(X) \leq V^i(X)$*

The proof can be found in the article [11]. So according to the theorem **2.2.2**, we only need an admissible initial control signal u^0, then the solutions of equations (2.17) and (2.18) will converge to the optimal solution: $\lim\limits_{i\to\infty} V^i = V^*$ and $\lim\limits_{i\to\infty} u^i = u^*$

2.2.2 Apply Data-driven PI to solve the above problem

From note **2.1.1**, the solution of the OTCP problem requires knowing both $f(x(t))$ and $g(x(t))$, there are many other methods that only need to know part of the model, such as $g(x(t))$, but the proposed algorithm in the following sections completely does not require model of the system. From the equation: $\dot{X} = F(X) + G(X)u$ we rewrite it as:

$$\dot{X} = F(X) + G(X)u^i + G(X)[u - u^i] \tag{2.19}$$

Where $u \in \mathbb{R}^m$. Differentiating $V^{i+1}(X)$ over time, combined with (2.17) and (2.18) we get:

$$\begin{aligned} \frac{dV^{i+1}(X)}{dt} &= [\nabla V^{i+1}]^T(F + Gu^i) + [\nabla V^{i+1}]^T G[u - u^i] \\ &= -X^T QX - [u^i]^T Ru^i + \lambda V^{i+1} + 2[u^{i+1}]^T R[u^i - u] \end{aligned} \tag{2.20}$$

Integrating both sides of (2.20) in $[t, t + \delta t]$ we get:

$$\begin{aligned} V^{i+1}(X(t+\delta t)) - V^{i+1}(X(t)) = &- \int_t^{t+\delta t} [X(\tau)^T QX(\tau) \\ &+ [u^i(X(\tau))]^T Ru^i(X(\tau))]d\tau \\ &+ \int_t^{t+\delta t} \lambda V^{i+1}(X(\tau))d\tau \\ &+ 2\int_t^{t+\delta t} [u^{i+1}(X(\tau))]^T R[u^i(X(\tau)) - u(\tau)]d\tau \end{aligned} \tag{2.21}$$

Looking at the above expression, we can see that $f(x(t))$ and $g(x(t))$ do not appear, so we do not need to know the dynamic model of the system. However, the system state X and the control signal u need to be known (need to be measured or calculated). Thus, the solution of OTCP can be obtained from the above equation.

Theorem 2.2.3. *If $V^{i+1}(X) \in \mathcal{V}(\mathcal{X})$, where $\mathcal{V}(\mathcal{X})$ is the function space on $\mathcal{X}$ is continuously differentiable, $V^{i+1}(X) \geq 0, V^{i+1}(0) = 0$ and $u^{i+1}(X) \in \mathcal{U}(\mathcal{X})$ then the solution of (2.21) is equivalent to the solution of (2.17) and (??)*

Proof: Appendix 1 According to the Weierstrass approximation theorem [5] a continuous function can be represented by an infinite set of linearly independent basis functions. Therefore, the Actor-Critic Neural Network (A-C NN) structure used to estimate the functions $V(X)$ and $u(X)$ on a closed set $\mathcal{X}$ is given as follows:

$$\begin{aligned} \hat{V}^i(X) &= [w_V^i]^T \varphi(X) \\ \hat{u}^i(X) &= [w_u^i]^T \psi(X) \end{aligned} \tag{2.22}$$

Here, $\varphi(X) \in \mathbb{R}^{l_\varphi}$ and $\psi(X) \in \mathbb{R}^{l_\psi}$ are 2 basis function vectors used to for approximation, respectively with the weights $w_V \in \mathbb{R}^{l_\varphi}$ and $w_u \in \mathbb{R}^{l_\psi \times m}$ (note $u \in \mathbb{R}^m$).
We define the estimation error $\epsilon_u^i(X) = \hat{u}^i(X) - u^i(X)$. The closed system with control law $\hat{u}^i(X)$ is rewritten as follows:

$$\dot{X} = F(X) + G(X)\hat{u}^i(X) \tag{2.23}$$

Theorem 2.2.4. *Consider the system (2.23) with approximate control law $\hat{u}^i(X)$, with estimation error $\epsilon_u^i(X)$ and decay factor λ approaches 0, the control signal $\hat{u}^i(X)$ will ensure that the tracking error $e_d(t)$ becomes asymptotically stable.*

Proof In the appendix
Now, let $t_{k-1} = t$ and $t_k = t + \delta t$. We consider the equation (2.21), for convenience in solving the Off-Policy equation, which can be rewritten as:

$$\begin{aligned} \sigma(X(t_{k-1}), u, X(t_k)) =& [\varphi(X(t_{k-1})) - \varphi(X(t_k))]^T w_V^{i+1} + \int_{t_{k-1}}^{t_k} \lambda [\varphi(X(\tau))]^T d\tau w_V^{i+1} \\ &+ 2\sum_{p=1}^{m} r_p \int_{t_{k-1}}^{t_k} [u_p^i - u_p(\tau)]^T \theta^T(X(\tau)) w_{u,p}^{i+1} d\tau \\ &- \int_{t_{k-1}}^{t_k} [X(\tau)^T Q X(\tau) d\tau - \int_{t_{k-1}}^{t_k} [u^i(\tau)]^T R u^i(\tau) d\tau \end{aligned} \tag{2.24}$$

$\sigma(X(t_{k-1}), u, X(t_k))$ is the error between the two sides of the equation (2.21)

$$\begin{aligned}
\xi_{\Delta\varphi k} &= \varphi(X(t_{k-1})) - \varphi(X(t_k)) \\
\xi_{Qk} &= \int_{t_{k-1}}^{t_k} [X(\tau)^T Q X(\tau) d\tau \\
\xi_{\lambda k} &= \int_{t_{k-1}}^{t_k} \lambda\varphi(X(\tau))) d\tau \\
\xi_{u\theta k,p} &= r_p \int_{t_{k-1}}^{t_k} [u_p^i - u_p(\tau)]^T \theta^T(X(\tau)) d\tau \\
\xi_{uk} &= \int_{t_{k-1}}^{t_k} [u^i(\tau)]^T R u^i(\tau) d\tau
\end{aligned} \tag{2.25}$$

From equations (2.24) and (2.25), it can be deduced:

$$\begin{aligned}
\sigma(X(t_{k-1}), u, X(t_k)) &= \xi_{\Delta\varphi k}^T w_V^{i+1} + \xi_{\lambda k} w_V^{i+1} \\
&= 2\sum_{p=1}^{m} \xi_{u\theta k,p} w_{u,p}^{i+1} - \xi_{Qk} - \xi_{uk} \\
&= \Lambda_k w_{Vu}^{i+1} - \Upsilon_k^i
\end{aligned} \tag{2.26}$$

Where:
$\Lambda_k = [(\xi_{\Delta\varphi k}^T + \xi_{\lambda k}^T), \xi_{u\theta k,p,1}, \xi_{u\theta k,p,m}]$
$\Upsilon_k^i = \xi_{Qk} + \xi_{uk}$
$w_{Vu}^{i+1} = [(w_V^{i+1})^T, (w_{u,1}^{i+1})^T ... (w_{u,m}^{i+1})^T]^T$

w_{Vu}^{i+1} can be obtained by minimizing $\sigma(X(t_{k-1}), u, X(t_k))$:

$$w_{Vu}^{i+1} = [\sum_{k=1}^{M} (\Lambda_k w_{Vu}^{i+1})^T (\Lambda_k w_{Vu}^{i+1})]^{-1} \sum_{k=1}^{M} (\Lambda_k w_{Vu}^{i+1})^T \Upsilon_k^i \tag{2.27}$$

M is the number of sampling intervals.
Note that the amount of data collected must be large enough and the additional noise u_e is important to satisfy the PE condition. In fact, in addition to adding noise to satisfy the PE condition, another way is to use many different control signals u to get data. It should also be noted that the upper bound condition of the decay factor λ, $\lambda \leq \bar{\lambda} = 2\|(B_1 R^{-1} B_1^T Q_e)^{1/2}\|$ to make the local error asymptotically stable, specifically see Appendix 3
Based on that, solving the equation (2.21) using the Data-driven method with M data collection time, we find the optimal solution. The algorithm is summarized as follows:

Algorithm 1. *Data-driven PI algorithm:*

---

1. *Initialize*
   Start with a stable control signal u^0 and add noise u_e to ensure the PE condition. Collect data and determine threshold ϵ

2. *Policy Evaluation*
   With the signal $u^i(X)$ solved from the previous loop, obtain $V^{i+1}(X)$ and $u^{i+1}(X)$ from the equation:

$$\begin{aligned} V^{i+1}(X(t+\delta t)) - V^{i+1}(X(t)) = &- \int_t^{t+\delta t} [X(\tau)^T Q X(\tau) \\ &+ [u^i(X(\tau))]^T R u^i(X(\tau))] d\tau \\ &+ \int_t^{t+\delta t} \lambda V^{i+1}(X(\tau)) d\tau \\ &+ 2 \int_t^{t+\delta t} [u^{i+1}(X(\tau))]^T R [u^i(X(\tau)) - u(\tau)] d\tau \end{aligned} \tag{2.28}$$

3. *Policy Improvement*
   Continue looping until $\|u^{n+1} - u^n\| < \epsilon$

2.3 Examples

2.3.1 The linear system

Consider a spring system with viscous friction described by the following equation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{k}{m} x_1 - \frac{c}{m} x_2 + \frac{1}{m} u(t) \end{aligned} \tag{2.29}$$

Where x_1 and x_2 are position and velocity respectively. $m = 1kg$, $c = 0.5Ns/m$ and $k = 5N/m$ The control trajectory that needs to be followed is:

$$\begin{aligned} x_{d1}(t) &= 0.5 sin(\sqrt{5}t) \\ x_{d2}(t) &= 0.5\sqrt{5} cos(\sqrt{5}t) \end{aligned} \tag{2.30}$$

It could be rewritten as:

$$\dot{x}_d = \begin{bmatrix} 0 & 1 \\ -5 & 0 \end{bmatrix} x_d \tag{2.31}$$

The expanded system is rewritten as follows:

$$\dot{X} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -5 & -0.5 & 0 & -0.5 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -5 & 0 \end{bmatrix} X + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u = AX + Bu \tag{2.32}$$

The parameters of the cost function can be chosen as follows: $R = 1$, $Q_e = 100I_2$ and $\lambda = 0.01$. The activation function for the basis function vector can be a Gaussian, Sigmoid or polynomial function. In this example we choose the polynomial as the activation function.
Critic's NN is selected as follows:

$$\begin{aligned} &\varphi(X) = [X_1^2, X_1X_2, X_1X_3, X_1X_4, X_2^2, X_2X_3, X_2X_4, X_3^2, X_3X_4, X_4^2]^T \\ &w_V = [w_{V1}..w_{V10}]^T \end{aligned} \tag{2.33}$$

Actor's NN is selected as follows:

$$\begin{aligned} \psi(X) = [&X_1, X_2, X_3, X_4, X_1^2, X_1X_2, X_1X_3, X_1X_4, X_2^2, X_2X_3, \\ &X_2X_4, X_3^2, X_3X_4, X_4^2, X_1^3, X_2^3, X_3^3, X_4^3]^T \\ w_u = [&w_{u1}..w_{u18}]^T \end{aligned} \tag{2.34}$$

Initial weight for the Actor:

$$\begin{aligned} w_u^{(0)} =&[0, 0, 0, 0.5, 0, 0, 0, 0, 0, 0, 0, \\ &0, 0, 0, 0, 0, 0, 0]^T \end{aligned} \tag{2.35}$$

Result after 24 iterations, weights converge:

$$\begin{aligned} w_V^{(24)} =&[115.118, 12.070, 4.477, 0.132, 10.211, 0.143, \\ &0.956, 61.910, 0.126, 12.382]^T \end{aligned} \tag{2.36}$$

$$\begin{aligned} w_u^{(24)} =&[6.135, 10.101, 0.0105, 0.477, 0, 0, 0, 0, 0, 0, 0, \\ &0, 0, 0, 0.00022, 0.00057, 0, 0]^T \end{aligned} \tag{2.37}$$

Figure (2.1) shows the actual trajectory x_1 and x_2 compared to the desired trajectory with the initial condition $x_0 = [-1, 1.5]^T$, using the Optimal controller with convergent weights. Since the system is linear, the solution to the problem, if the dynamic model of the system is known, can be obtained from solving the ARE equation below:

$$A^TP + PA - \lambda P + PBR^-1B^TP + Q = 0 \tag{2.38}$$

$$P^* = \begin{bmatrix} 115.5458 & 6.1266 & -2.2452 & -0.0249 \\ 6.1266 & 10.1005 & 0.0132 & -0.4771 \\ -2.2452 & 0.0132 & 62.3598 & -0.0622 \\ -0.0249 & -0.4771 & -0.0622 & 12.4730 \end{bmatrix}$$

From the matrix $P*$, we obtain w_V^*:

$$\begin{aligned} w_V^* =&[115.5458, 12.2532, 4.4904, 0.0498, 10.1005, 0.0264, \\ &0.9542, 62.3598, 0.1244, 12.4730]^T \end{aligned}$$

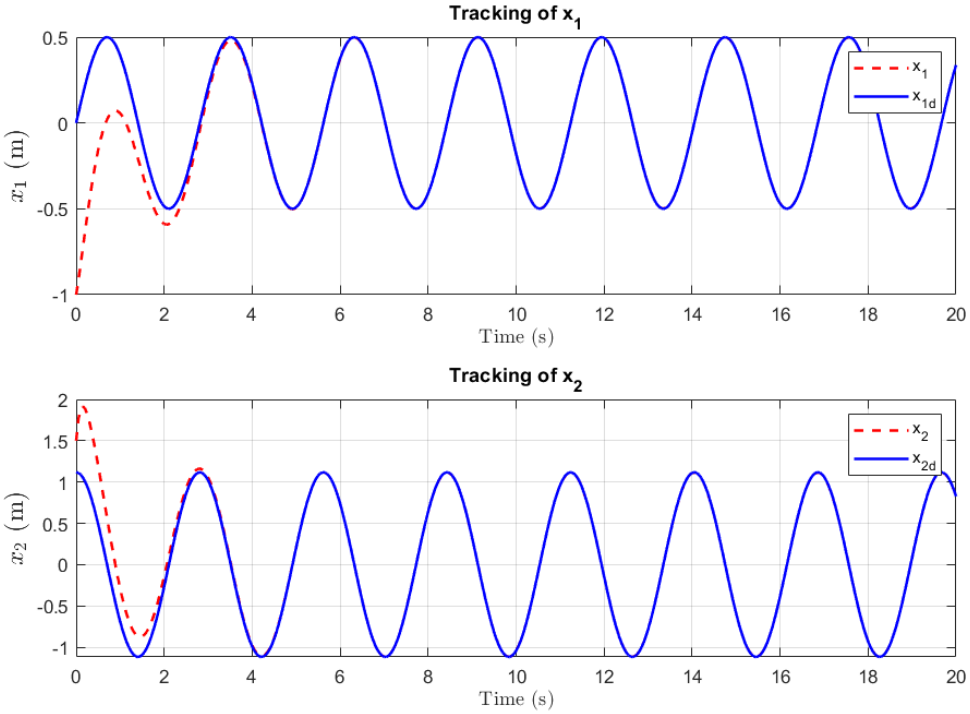


Figure 2.1: The tracking trajectory of the linear system

Comparing w_V^* and $w_V^{(24)}$, we see that the results are quite similar, proving the correctness of the algorithm
To verify the effect of λ on control quality, keep the original parameters unchanged, change λ to 0.01, 0.1 and 0.5 respectively, we obtain the graph in the figure (refFig:3.2)

Notation 2.3.1. *The upper bound of the error e_1 are 0.0001, 0.00027 and 0.02 corresponding to the λ values of 0.01, 0.1 and 0.5. It can be seen that the error can be adjusted as small as desired by choosing a sufficiently small decay factor λ.*

2.3.2 The nonlinear system

Consider the affine nonlinear system:

$$\dot{x} = \begin{bmatrix} x_2 \\ -0.5(x_1 + x_2) + 0.5x_1^2 x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{2.39}$$

Desired trajectory $x_d = [0.5sin(t), 0.5cos(t)]^T$, therefore:

$$\dot{x}_d = \begin{bmatrix} x_{2d} \\ -x_{1d} \end{bmatrix}$$

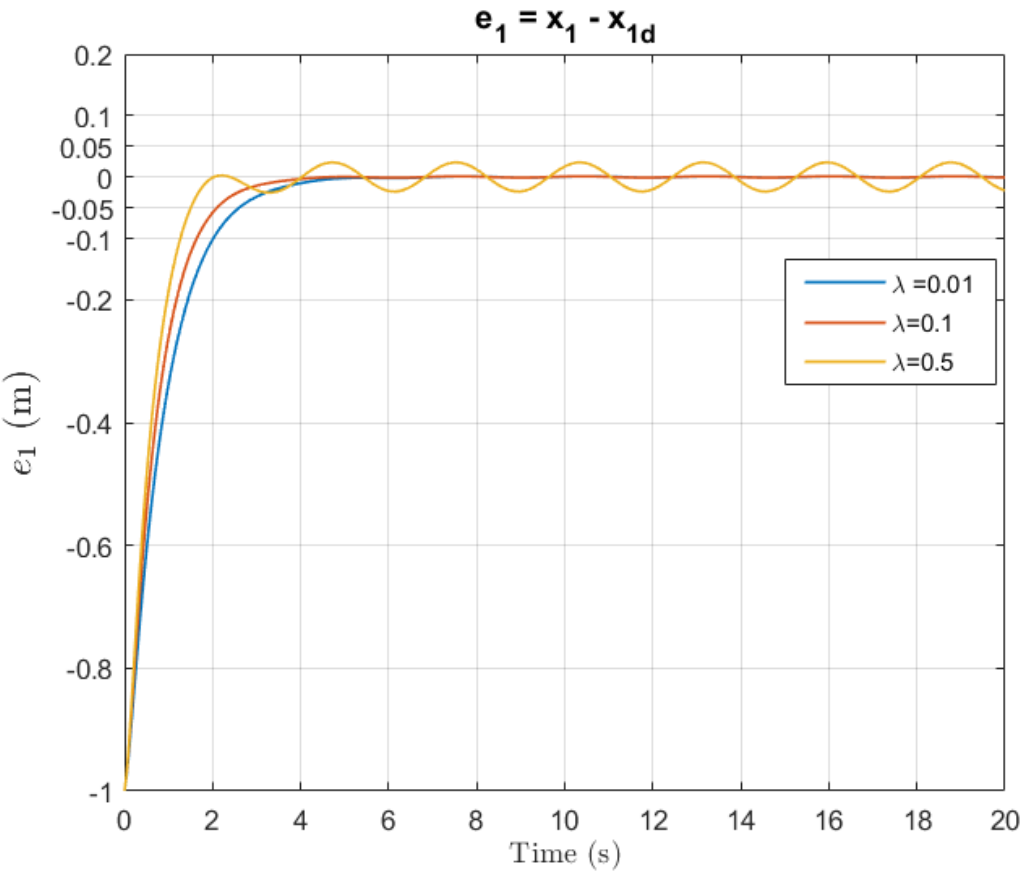


Figure 2.2: e_1 depends on λ

Initial conditions $x(0) = [0.2\ 0.1]^T$ and $x_d(0) = [0\ 0.5]^T$. The cost function parameters are chosen with $Q_e = 100I_2$ and $R = 1$, $\lambda = 0.01$. Critic NN selected is:

$$\begin{aligned}\varphi(X) = [&X_1^2, X_1X_2, X_1X_3, X_1X_4, X_2^2, X_2X_3, X_2X_4, X_3^2,\\ &X_3X_4, X_4^2, X_1^3, X_2^3, X_3^3, X_4^3, X_1^4, X_2^4, X_3^4, X_4^4]^T\\ w_V = [&w_{V1}...w_{V18}]^T\end{aligned} \tag{2.40}$$

Actor NN selected is:

$$\begin{aligned}\psi(X) = [&X_1, X_2, X_3, X_4, X_1^2, X_1X_2, X_1X_3, X_1X_4, X_2^2, X_2X_3,\\ &X_2X_4, X_3^2, X_3X_4, X_4^2, X_1^3, X_1X_2X_3, X_1X_2X_4, X_1^2X_2,\\ &X_1^2X_3, X_1^2X_4, X_2^3, X_2^2X_1, X_2^2X_3, X_2^2X_4, X_2X_3X_4,\\ &X_3^3, X_3^2X_1, X_3^2X_2, X_3^2X_4, X_4^3, X_4^2X_1, X_4^2X_2, X_4^2X_3]^T\\ w_u = [&w_{u1}...w_{u34}]^T\end{aligned} \tag{2.41}$$

The initial weight of the Actor is chosen as follows:

$$\begin{aligned}w_u^{(0)} =&[0, 0, -0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0.5,\\ &0, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0, 0]^T\end{aligned} \tag{2.42}$$

After 60 iterations, the weights converge to the following result:

$$\begin{aligned}w_V^{(60)} = [&102.7629, 18.9965, 1.1704, 0.3302, 10.6051, 1.0414, -0.7600, 22.6232, -0.0341,\\ &22.4059, 8.6455, 0.9452, 0.0021, 0.0006, 781.7642, 1.2293, -0.3745, 0]^T\end{aligned}$$

$$\begin{aligned}w_u^{(60)} = [&-7.7546, -10.9013, -0.5733, 0.4068, 33.2184, 6.4724, -2.2191, 4.4119,\\ &-0.5362, -0.1452, 0.5718, 0.0333, -0.1610, 0.1357, -432.6140, -0.9517,\\ &51.7376, -0.6381, -143.7584, -30.0475, -121.1779, -7.1010, -41.0103, 10.2111,\\ &-9.9278, -1.2006, 0, 0, 0, 0.4219, 0, -10.4996,\\ &-0.7099, 0.3058]^T\end{aligned}$$

Using the optimal controller with the above set of convergent weights, we obtain the results as shown in Figure 2.3. We can see that the tracking control quality is quite good, the tracking error converges to 0 after about 4 (s).
So through 2 examples of applying the Data-driven PI algorithm to 2 quite simple linear and nonlinear systems, we can see the correctness of the algorithm. To demonstrate the applicability to complex nonlinear systems, the next chapter studies the application of the algorithm to find the optimal controller for the unmanned aircraft "Quadrotor".

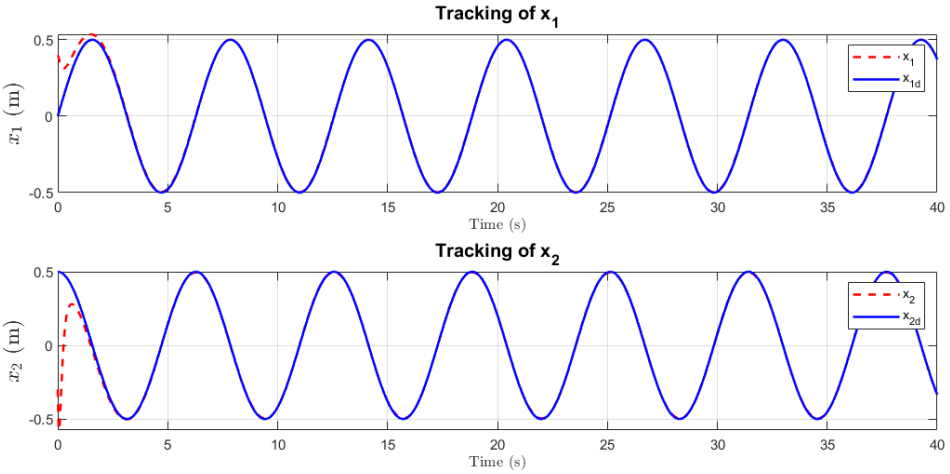
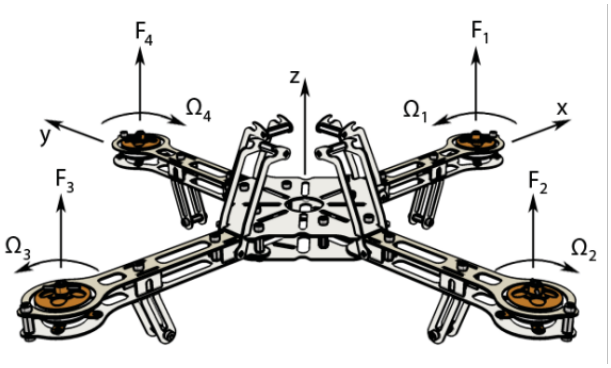


Figure 2.3: The tracking trajectory of the nonlinear system

Chapter 3

Apply Data-Driven PI algorithm for quadrotor

3.1 Introduction to Quadrotors



In recent years, drones have received great attention in the research community thanks to their great potential for application in many jobs and fields where human presence is limited: disaster survey, forest fire detection, agricultural applications,...

One of the typical types of drones, quadrotors have become increasingly popular thanks to their ability to take off and land vertically, position stability and flexible orbits.

Accordingly, one of the most popular and important problems raised is the problem of tracking control for quadrotors. This is not a simple problem because the quadrotor has 6 degrees of freedom with 4 control inputs, and the quadrotor model is also an object model with heavy coupling. Moreover, in practical problems and applications, knowing the exact object model is difficult, accompanied by model uncertainty that is often encountered due to the frequent task requirements of the quadrotor: attaching a complex heavy object, for example: a camera, an item to be transported in a delivery application, etc. Therefore, the requirement to resolve model uncertainty is an important requirement when designing a controller for quadrotors. In many recent studies, to achieve good control quality, many control structures have been proposed, including: PID controller [6],[18], LQR controller citearticle10, [19], backstepping controller [3], [4], [12], sliding mode controller [15]-[20]. In [19], the LQR controller is designed to stabilize the quadrotor model

in a small working range. In [15], the fast nonlinear sliding mode control structure is studied with the aim of bringing the error variable to the equilibrium position within a finite period of time. Besides, many observer-based methods are also used to compensate kinematic errors for the ideal quadrotor model, then a nonlinear feedback controller is designed to solve the tracking control problem. .
In this section, we apply the Data-driven PI algorithm presented in the previous sections to the quadrotor object.

3.2 Quadrotor kinematic model

Quadrotor can be simply described by the picture below:

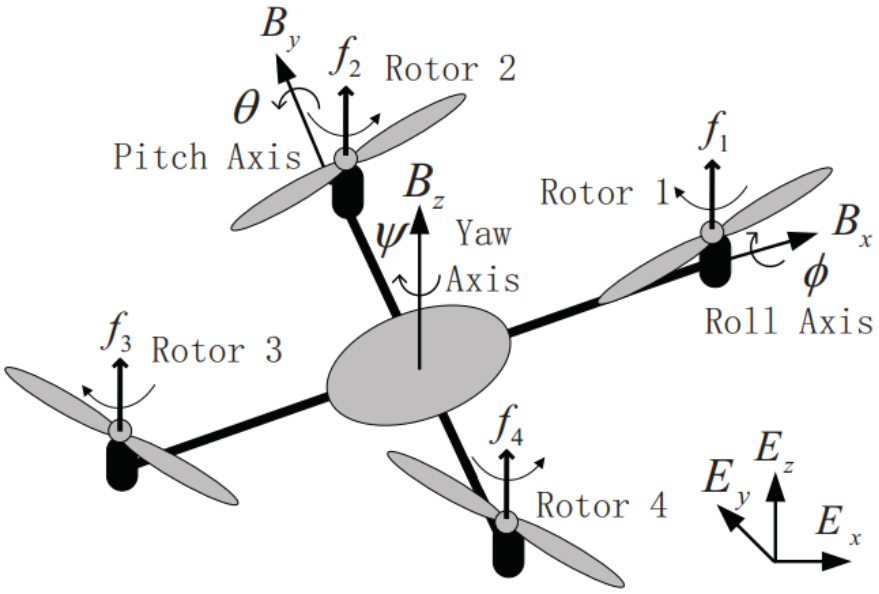


Figure 3.1: The kinematics of a quadrotor

Earth-fixed inertial frame(Earth-fixed inertial frame): is a fixed coordinate frame attached to the Earth, denoted: $\alpha = E_x E_y E_z$ (as shown in figure 4.1)
Body-fixed frame: is a coordinate frame with the origin attached to the center of mass of the quadrotor, describing the direction of rotation of the quadrotor compared to the coordinate frame attached to the Earth, symbol: $\beta = B_x B_y B_z$ (as shown in figure 4.1)
Symbol:
The position of the quadrotor center of mass or origin of the β frame relative to the α coordinate frame: $p = [p_x, p_y, p_z]^T \in \mathbb{R}^3$. The Euler angles relative to the α coordinate frame are $\Theta = [\phi, \theta, \psi] \in \mathbb{R}^3$.
The angular velocity of the quadrotor in the β reference frame is $\omega = [p, q, r]^T$,

this angular velocity is different from the time derivative of the Euler angles $\dot{\Theta} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]$ because Euler angles are ordered rotation angles, the formula relating them is:

$$\omega = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \dot{\Theta} \tag{3.1}$$

The Euler angle ZXY is used to describe the rotation of the quadrotor in a coordinate system attached to the Earth, $R = R_{\beta\to\alpha} = R_{Z,\psi}R_{Y,\theta}R_{X,\phi}$ is the unit conversion matrix from coordinates β to α:

$$R_{Z,\psi} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \; R_{Y,\theta} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \; R_{X,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}$$

$$R = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{3.2}$$

Dynamic equation:

$$m\ddot{p} = T_p R e_{3,3} - mge_{3,3} \tag{3.3}$$

$$J\dot{\omega} = \tau - \omega \times (J\omega) \tag{3.4}$$

Here $T_p \in \mathbb{R}$ is the total lift force of the blades $T_p = T_1 + T_2 + T_3 + T_4$ in the coordinate system α and $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^T \in \mathbb{R}^3$ is the moment acting on the quadrotor in the β coordinate system rotating around the x, y, z axes, $T_p = k_w u_z$ and $\tau = [l_\tau k_w u_\phi,\ l_\tau k_w u_\theta,\ k_t u_\psi]^T$. m is the mass of the quadrotor, g is the gravitational acceleration, $J = diag(J_x, J_y, J_z)$ with $J_x,\ J_y,\ J_z$ respectively the moments of inertia around the x, y, z axes respectively. It can be seen from the quadrotor's dynamic model that there are 6 DOFs with 4 inputs and is a super-coupled nonlinear system. In fact, the input signals $u_z, u_\phi, u_\theta, u_\psi$ depend on the rotation speed of the blades as follows:

$$\begin{aligned} u_z &= \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 \\ u_\phi &= \omega_2^2 - \omega_4^2 \\ u_\theta &= \omega_1^2 - \omega_3^2 \\ u_\psi &= \omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2 \end{aligned} \tag{3.5}$$

Where $\omega_j (j = 1, 2, 3, 4)$ are the rotational velocities of the corresponding j blades.

In addition, based on the relation in the formula (3.1) and (3.4), we obtain

another system of dynamic equations:

$$\begin{aligned} m\ddot{p} &= T_p R e_{3,3} - mge_{3,3} \\ J\ddot{\Theta} &= \tau - C(\Theta, \dot{\Theta})\dot{\Theta} \end{aligned} \tag{3.6}$$

Ở đây:

$$C(\Theta, \dot{\Theta}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

$$\begin{aligned}
c_{11} &= 0 \\
c_{12} &= (J_y - J_z)(\dot{\theta}c\phi s\phi + \dot{\psi}s^2\phi c\theta)/J_x + ((J_z - J_y)\dot{\psi}c^2\phi c\theta - J_x\dot{\psi}c\theta)/J_x, \\
c_{13} &= (J_z - J_y)\dot{\psi}c\phi s\phi c^2\theta/J_x, \\
c_{21} &= (J_z - J_y)(\dot{\theta}c\phi s\phi + \dot{\psi}s^2\phi c\theta)/J_y + ((J_y - J_z)\dot{\psi}c^2\phi c\theta + J_x\dot{\psi}c\theta)/J_y, \\
c_{22} &= (J_z - J_y)\dot{\phi}c\phi s\phi/J_y, \\
c_{23} &= (-J_x\dot{\psi}s\theta c\theta + J_z\dot{\psi}c^2\phi c\theta s\theta)/J_y + \dot{\psi}s^2\phi c\theta s\theta, \\
c_{31} &= ((J_y - J_z)\dot{\psi}c^2\theta s\phi c\phi - J_x\dot{\theta}c\theta)/J_z, \\
c_{32} &= (J_z - J_y)(\dot{\theta}c\phi s\phi s\theta + \dot{\phi}s^2\phi c\theta)/J_z + ((J_y - J_z)\dot{\phi}c^2\phi c\theta + J_x\dot{\psi}s\theta c\theta)/J_z \\
&\quad - J_y\dot{\psi}s^2\phi s\theta c\theta/J_z - \dot{\psi}c^2\phi c\theta s\theta, \\
c_{33} &= (-J_z\dot{\theta}c^2\phi s\theta c\theta - J_y\dot{\theta}s^2\phi c\theta s\theta)/J_z + ((J_y - J_z)\dot{\phi}c\phi s\phi c^2\theta + J_x\dot{\theta}s\theta c\theta)/J_z
\end{aligned}$$

The formula (3.6) is mainly used in this project

3.3 The typical control scheme for a quadrotor

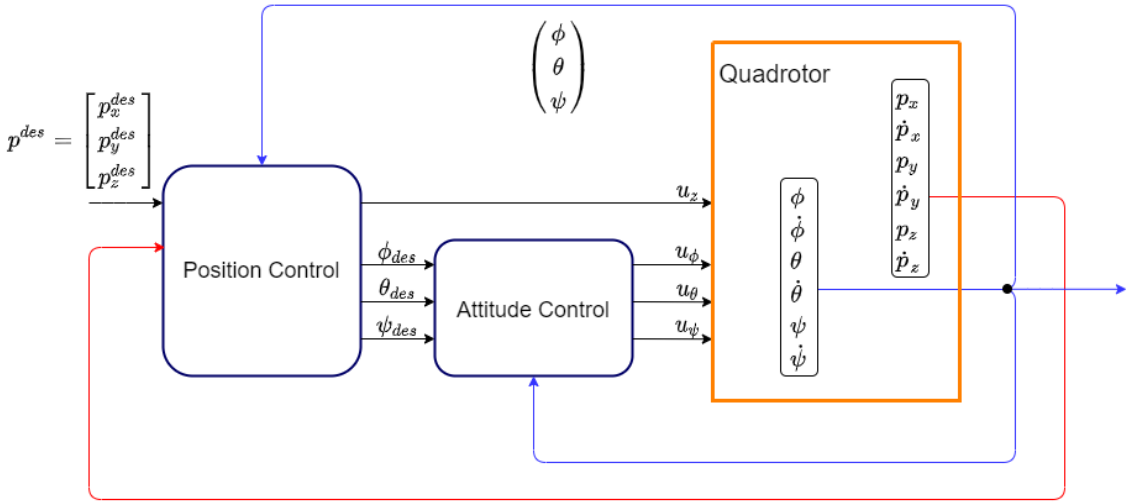


Figure 3.2: The typical control scheme for a quadrotor

The picture above is the principle of controlling a Quadrotor in general, because the quadrotor is a nonlinear system, many components are coupled but in general they can be separated into 2 main controllers:

1. *Position Controller*: generates a set angle trajectory for the Attitude Controller.

2. *Attitude Controller*: tracks the set angle trajectory generated by the Position Controller

It can be simply understood that the purpose of control is to adjust the rotation speed of the 4 motors so that the quadrotor can tilt in the right direction, combined with the lifting force to track the desired trajectory. The feedback measurement signals are very important, for the quadrotor model one can completely measure all of its physical quantities including:

1. *Position, velocity and acceleration*: In research in laboratories (Indoor), the Vicon System is used to accurately measure the above quantities .

2. *Euler angle, Angular velocity and angular acceleration*: Using familiar IMU (Internal Measuring Unit) sensors also helps measure the above quantities

3. In addition, in the case of outdoor aircraft, they often add a pressure sensor (barometer), measure position using GPS (Global Positioning System) and save parameters via black box.

In this project, we study quadrotor simulation at the laboratory level (Indoor), so most of the feedback quantities can be measured.

3.4 Data-driven PI algorithm for Optimal tracking control problem of quadrotor with complete unknown model

3.4.1 Position Control with Data-driven PI

The equation (3.3) can be rewritten as:

$$\begin{aligned} \ddot{p} &= m^{-1} k_w u_z R e_{3,3} - g e_{3,3} \\ &= m^{-1} k_w u_p \end{aligned} \tag{3.7}$$

Here: $u_p = u_z R e_{3,3} - \frac{m}{k_w} g e_{3,3} \in \mathbb{R}^3$, off-set component $u_b = \ fracmk_w ge_{3,3}$ can in fact be completely determined without knowing m, g, k_w.Appendix
Let $x_p = [p_x, \dot{p}_x, p_y, \dot{p}_y, p_z, \dot{p}_z]^T \in \mathbb{R}$, equation (**??**) can be rewritten as follows:

$$\dot{x}_p = A_p x_p + B_p u_p \tag{3.8}$$

With $A_p = diag(a_p, a_p, a_p) \in \mathbb{R}^{6\times 6}$, $a_p = [0_{2\times 1}\ e_{2,1}]$ and $B_p = m^{-1} k_w [e_{6,2}, e_{6,4}, e_{6,6}]$
Suppose the desired trajectory is x_{pd} with equation $\dot{x}_{pd} = A_{pd} x_{pd}$, and error

$e_p = x_p - x_{pd}$, the expanded equation can be written as:

$$\dot{X}_p = \begin{bmatrix} \dot{e}_p \\ \dot{x}_{pd} \end{bmatrix} = \begin{bmatrix} A_p & A_p - A_{pd} \\ 0_{6\times6} & A_{pd} \end{bmatrix} X_p + \begin{bmatrix} B_p \\ 0_{6\times3} \end{bmatrix} u_p \tag{3.9}$$

Matrix $Q_p = \begin{bmatrix} Q_{ep} & 0_{6\times6} \\ 0_{6\times6} & 0_{6\times6} \end{bmatrix}$, where Q_{ep} is a positive definite symmetric matrix, $R_p \in \mathbb{R}^{3\times3}$ is also a positive definite symmetric matrix.
With the cost function chosen as:

$$V_p(X_p(t)) = \int_t^\infty e^{-\lambda(\tau-t)}(X_p(\tau)^T Q_p X_p(\tau) + u_p(\tau)^T R_p u_p(\tau))d\tau \tag{3.10}$$

As presented in chapter 2, similarly we have a Data-driven PI algorithm for Position controller as follows:

Algorithm 2. *Data-driven PI for Position controller:*

---

1. *Initialization*
   Start with a stable control signal u_p^0 and add noise u_{pe} to ensure the PE condition. Collect data and determine threshold ϵ_p

2. *Policy Evaluation*
   With the signal $u_p^i(X_p)$ obtained from the previous loop, solve $V_p^{i+1}(X_p)$ and $u_p^{i+1}(X_p)$ from the equation:

$$\begin{aligned} V_p^{i+1}(X_p(t+\delta t)) - V_p^{i+1}(X_p(t)) = &- \int_t^{t+\delta t} [X_p(\tau)^T Q_p X_p(\tau) \\ &+ [u_p^i(X_p(\tau))]^T R_p u_p^i(X_p(\tau))]d\tau \\ &+ \int_t^{t+\delta t} \lambda V_p^{i+1}(X_p(\tau))d\tau \\ &+ 2\int_t^{t+\delta t} [u_p^{i+1}(X_p(\tau))]^T R_p u_p^i(X_p(\tau))d\tau \\ &- 2\int_t^{t+\delta t} [u_p^{i+1}(X_p(\tau))]^T R_p [u_p^0(\tau) + u_{pe}]d\tau \end{aligned} \tag{3.11}$$

3. *Policy Improvement*
   Continue until $\|u_p^{i+1} - u_p^i\| < \epsilon_p$

To approximate V_p^i and u_p^i, Critic-Actor NN is constructed as follows:

$$V_p^i(X_p) = w_{Vp}^T \varphi_p(X_p) \tag{3.12}$$

$$u_p^i(X_p) = w_{up}^T \psi_p(X_p) \tag{3.13}$$

Here $\varphi_p(X_p) \in \mathbb{R}^{l_1}$ and $\psi_p(X_p) \in \mathbb{R}^{l_2}$ are two basis function vectors (l_1 and l_2 are the number of neurons of $\varphi_p(X_p)$ and $\psi_p(X_p)$, respectively).$w_{Vp} \in \mathbb{R}^{l_1 \times 1}$ and $w_{up} \in \mathbb{R}^{l_2 \times 3}$ are the two corresponding weight vectors. The Least-Square method can be applied to solve the equation (3.11).
After finding the optimal signal $u_p = [u_{px}, u_{py}, u_{pz}]^T$, from (3.7) we can solve the control signal u_z, the desired angle trajectory for the inner control loop is as follows (the Yaw angle ψ_d is chosen fixedly):

$$\begin{aligned} u_z &= \sqrt{u_{px}^2 + u_{py}^2 + (u_{pz} + u_b)^2} \\ \psi_d &= 0 \\ \phi_d &= arcsin(\frac{u_{px} sin(\psi_d) - u_{py} cos(\psi_d)}{u_z}) \\ \theta_d &= arctan(\frac{u_{px} cos(\psi_d) + u_{py} sin(\psi_d)}{u_{pz} + u_b}) \end{aligned} \tag{3.14}$$

3.4.2 Attitude Control with Data-driven PI

Define $x_\Theta = [\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}]^T$ equation (**??**) we can rewrite it as:

$$\dot{x}_\Theta = F_\Theta x_\Theta + B_\Theta u_\Theta \tag{3.15}$$

Ở đây $F_\Theta = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -c_{11}J_x^{-1} & 0 & -c_{12} & 0 & -c_{13} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -c_{21} & 0 & -c_{22}J_y^{-1} & 0 & -c_{23} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -c_{31} & 0 & -c_{32} & 0 & -c_{33}J_z^{-1} \end{bmatrix}$

$B_\Theta = [e_{6,2}b_{\Theta 1}, e_{6,4}b_{\Theta 2}, e_{6,6}b_{\Theta 3}] \in \mathbb{R}^{6\times 3}$
$b_{\Theta 1} = J_x^{-1} l_\tau k_w$, $b_{\Theta 2} = J_y^{-1} l_\tau k_w$, $b_{\Theta 3} = J_z^{-1} k_t$.
The desired trajectory from the formula (3.14) can be written as $\dot{x}_{\Theta d} = F_{\Theta d} x_{\Theta d}$ and the attitude error $e_\Theta = x_\Theta - x_{\Theta d}$, the expanded equation can be rewritten as:

$$\dot{X}_{\Theta d} = \begin{bmatrix} \dot{e}_\Theta \\ \dot{x}_{\Theta d} \end{bmatrix} = \begin{bmatrix} F_\Theta & F_\Theta - F_{\Theta d} \\ 0_{6\times 6} & F_{\Theta d} \end{bmatrix} X_{\Theta d} + \begin{bmatrix} B_\Theta \\ 0_{6\times 3} \end{bmatrix} u_\Theta \tag{3.16}$$

Matrix $Q_\Theta = \begin{bmatrix} Q_{e\Theta} & 0_{6\times 6} \\ 0_{6\times 6} & 0_{6\times 6} \end{bmatrix}$, where $Q_{e\Theta}$ is a positive definite symmetric matrix, $R_\Theta \in \mathbb{R}^{3\times 3}$ is also a positive definite symmetric matrix. With the cost function

chosen as:

$$V_\Theta(X_\Theta(t)) = \int_t^\infty e^{-\lambda(\tau-t)}(X_\Theta(\tau)^T Q_e X_\Theta(\tau) + u_\Theta(\tau)^T R u_\Theta(\tau))d\tau \tag{3.17}$$

As presented in chapter 2, similarly we have a Data-driven PI algorithm for Attitude Controller as follows:

Algorithm 3. *Data-driven PI for Attitude Controller:*

---

1. *Khởi tạo*
   Start with a stable control signal u_Θ^0 *and add noise* $u_{\Theta e}$ *to ensure the PE condition. Collect data and determine threshold* ϵ_Θ

2. *Policy Evaluation*
   With the signal $u_\Theta^i(X_\Theta)$ *obtained from the previous loop, solve* $V_\Theta^{i+1}(X_\Theta)$ *and* $u_\Theta^{i+1}(X_\Theta)$ *from the equation:*

$$\begin{aligned} V_\Theta^{i+1}(X_\Theta(t+\delta t)) - V_\Theta^{i+1}(X_\Theta(t)) = &- \int_t^{t+\delta t} [X_\Theta(\tau)^T Q_\Theta X_\Theta(\tau) \\ &+ [u_\Theta^i(X_\Theta(\tau))]^T R_\Theta u_\Theta^i(X_\Theta(\tau))]d\tau \\ &+ \int_t^{t+\delta t} \lambda V_\Theta^{i+1}(X_\Theta(\tau))d\tau \\ &+ 2\int_t^{t+\delta t} [u_\Theta^{i+1}(X_\Theta(\tau))]^T R_\Theta u_\Theta^i(X_\Theta(\tau))d\tau \\ &- 2\int_t^{t+\delta t} [u_\Theta^{i+1}(X_\Theta(\tau))]^T R_\Theta [u_\Theta^0(\tau) + u_{\Theta e}]d\tau \end{aligned} \tag{3.18}$$

3. *Policy Improvement*
   Continue until $\|u_\Theta^{i+1} - u_\Theta^i\| < \epsilon_\Theta$

To approximate V_Θ^i and u_Θ^i, Critic-Actor NN is constructed as follows:

$$V_\Theta^i(X_\Theta) = w_{V\Theta}^T \varphi_\Theta(X_\Theta) \tag{3.19}$$

$$u_\Theta^i(X_\Theta) = w_{u\Theta}^T \psi_\Theta(X_\Theta) \tag{3.20}$$

Here $\varphi_\Theta(X_\Theta) \in \mathbb{R}^{l_3}$ and $\psi_\Theta(X_\Theta) \in \mathbb{R}^{l_4}$ are 2 basis function vectors (l_3 and l_4 are the number of neurons of $\varphi_\Theta(X_\Theta)$ and $\psi_\Theta(X_\Theta)$, respectively).$w_{V\Theta}$ $in\mathbb{R}^{l_3\times 1}$ and $w_{u\Theta} \in \mathbb{R}^{l_4\times 3}$ are two corresponding weight vectors. The Least-Square method can be applied to solve the equation (3.18).

3.5 The simulation results

Consider the object with the following parameters:
$m = 2.0\,kg, k_w = 1, k_t = 1, g = 9.8\,m/s^2, l_\tau = 0.2\,m$

$$J = \begin{bmatrix} 5.1 & 0 & 0 \\ 0 & 5.1 & 0 \\ 0 & 0 & 10.2 \end{bmatrix} \quad (10^{-3} kg.m^2)$$

The desired trajectory in the simulation is chosen as the spiral orbit $p_d = [2sin(at),\ 2cos(at),\ 0.8t]^T$ with $a = 0.5$ as the angular frequency.

First, in the data collection stage, we use 2 simple PD (Proportion-Derivative) controllers for both Position and Attitude control loops to keep the system stable in the beginning. The coefficient $[K_p,\ K_d]$ is chosen empirically with [12, 15] for the outer loop and [5, 100] for the inner loop.
In addition, to ensure the PE condition, noise signals are added to the two controllers, specifically $u_{pe} = \sum_{m=1}^{100} 0.01 sin(w_m t)$ with the position control loop, and $u_{\Theta e} = \sum_{m=1}^{500} 0.002 sin(w_m t)$, where w_m are randomly selected frequencies in the range $[-100, 100]$.
With the original controller used to collect data, we have the tracking error of position and direction angle as follows (Figure (3.3) and (3.4))

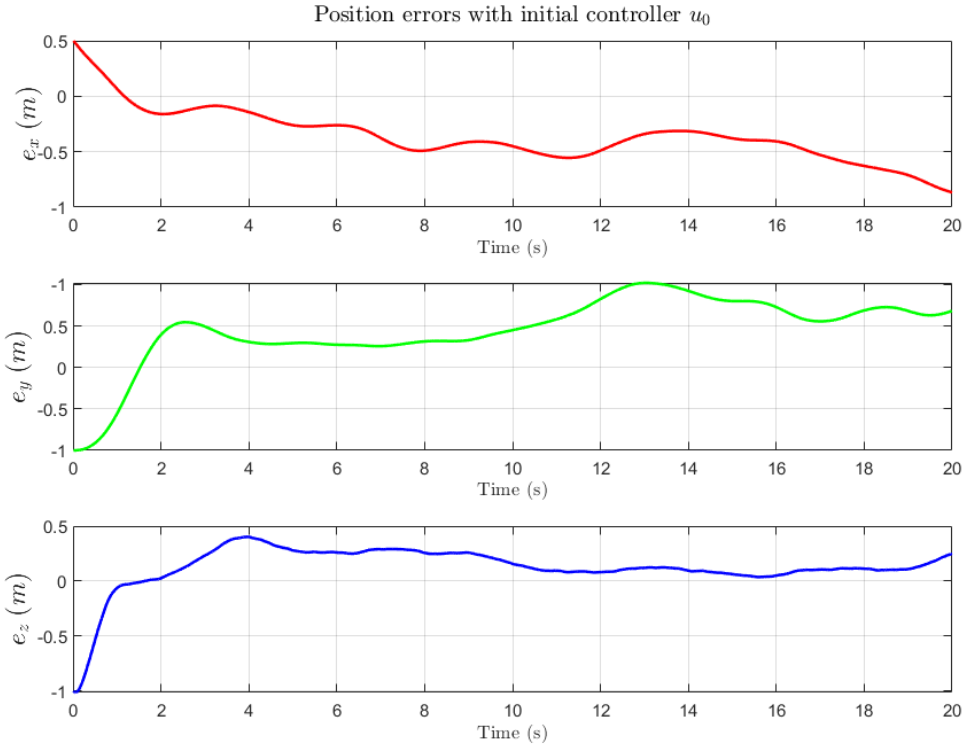


Figure 3.3: Position tracking errors with original controller

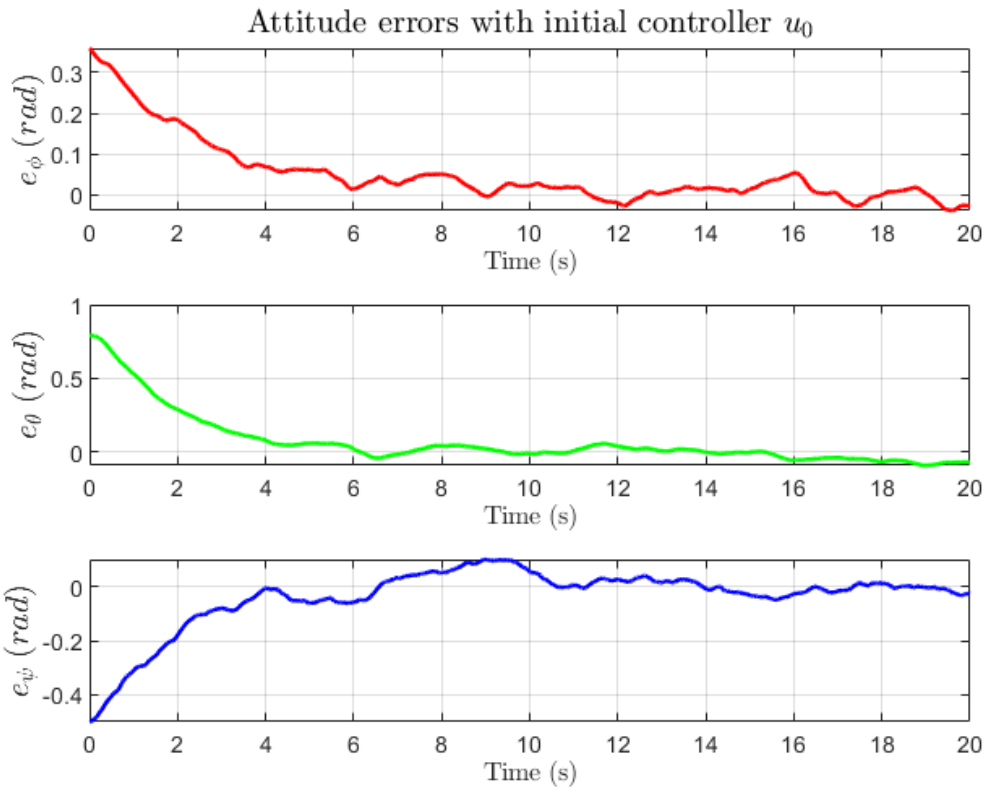


Figure 3.4: Attitude tracking error with original controller

After collecting data, apply two algorithms (3.11) for the Position controller and (3.18) for the Attitude controller with the following parameters:

For the Position controller, choose the matrix $Q_e = 100I_6$, $R_p = I_3$ and the coefficient $\lambda = 0.01$
For the Attitude controller, choose the matrix $Q_{\Theta e} = 100I_6$, $R_\Theta = I_3$ and the coefficient $\lambda = 0.01$
For the basis function vector chosen in both algorithms, the Actor's basis function vector is a first-order function, the Critic's basis function vector is a second-order function.
The integration time chosen in both algorithms is $T_{step} = 0.01$
The results after training and weight convergence are shown in figures (3.5) and (3.6).

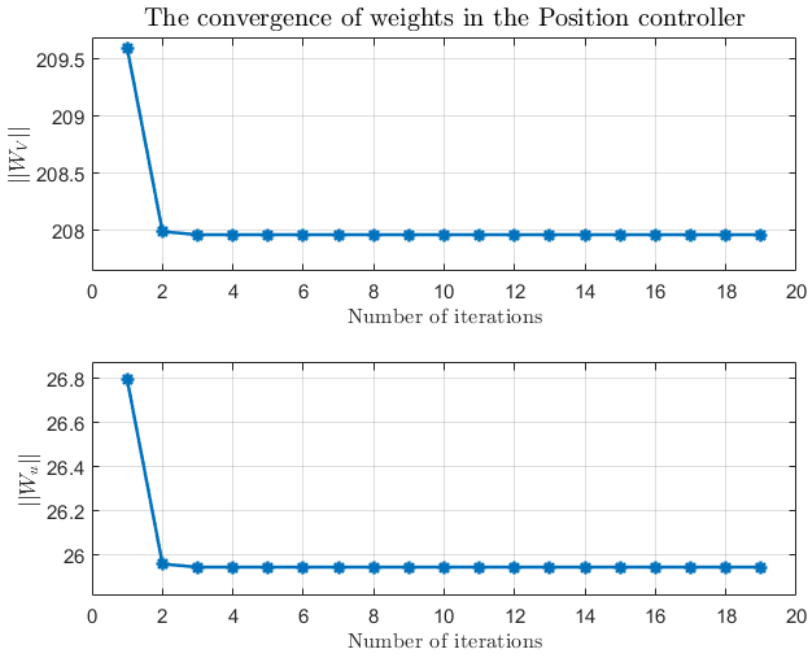


Figure 3.5: Convergence of weights' norm in Position controller

In the Position controller, the weights of Actor and Critic converge after 4 iterations. With the Attiude controller, convergence is achieved after 8 iterations. We can see that the weights converge quite quickly in both controllers Specifically, the Actor's convergence weights at the position and direction controllers are:

$$W_{up}^{*} = \begin{bmatrix} -9.4257 & 0 & 0 \\ -11.6351 & 0 & 0 \\ 0 & -9.4257 & 0 \\ 0 & -11.6351 & 0 \\ 0 & 0 & -9.4257 \\ 0 & 0 & -11.6351 \\ -0.5119 & 0.0199 & 0 \\ 0 & 0 & 0 \\ -0.0199 & -0.5119 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -0.0053 \end{bmatrix}$$

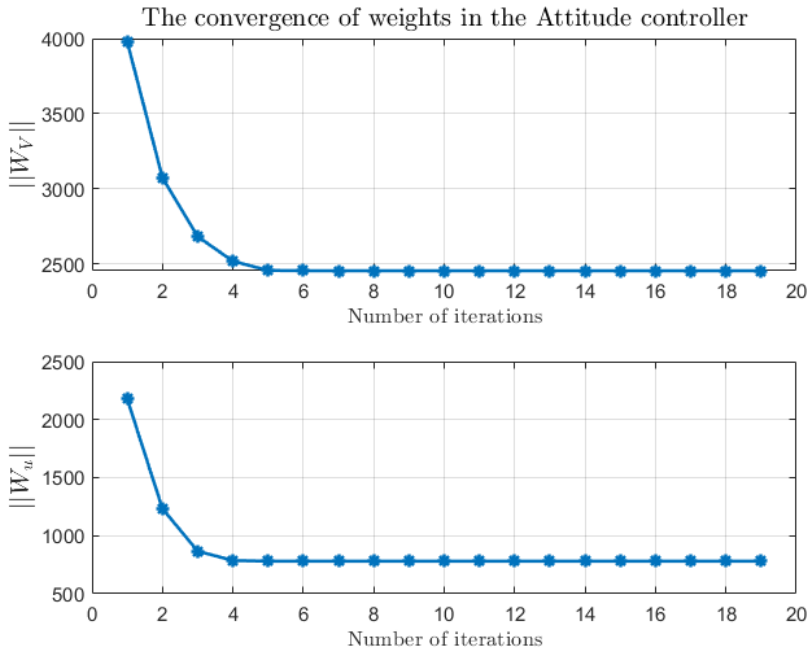


Figure 3.6: Convergence of weights' norm in Attitude controller

$$W^*_{u\Theta} = \begin{bmatrix} -299.9024 & 0.9506 & 4.6424 \\ -338.1839 & 1.0954 & 1.3983 \\ -3.8254 & -298.8637 & -0.9325 \\ 1.1142 & -338.8784 & 0.4628 \\ -0.3346 & -0.6826 & -301.0987 \\ 0.1909 & 0.2886 & -324.5006 \\ 12.1182 & -22.1137 & -12.5393 \\ -14.4522 & -19.2389 & 3.7538 \\ 7.7199 & 2.0119 & -2.7786 \\ 34.3998 & -45.8008 & -22.3954 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Using the Actor's convergence weight, applying control to the original system, we get the results as shown (3.7) and (3.8). From (3.7) and (3.8), we can see that the trajectory tracking quality is quite good, the position tracking error is approximately 0 after about 5 s and the attitude angle error is approximately 0 after about 6 s. To see the control quality more clearly, the figure (3.9) shows the position tracking graph of the three axes x, y, z and the figure shows the 3D trajectory with the optimal controller.

To verify the effect of λ on control quality, the simulation was conducted in exactly the same way as above, with one parameter changing $\lambda = 0.5$ and $\lambda = 1$. The results are shown in the figure (3.10)

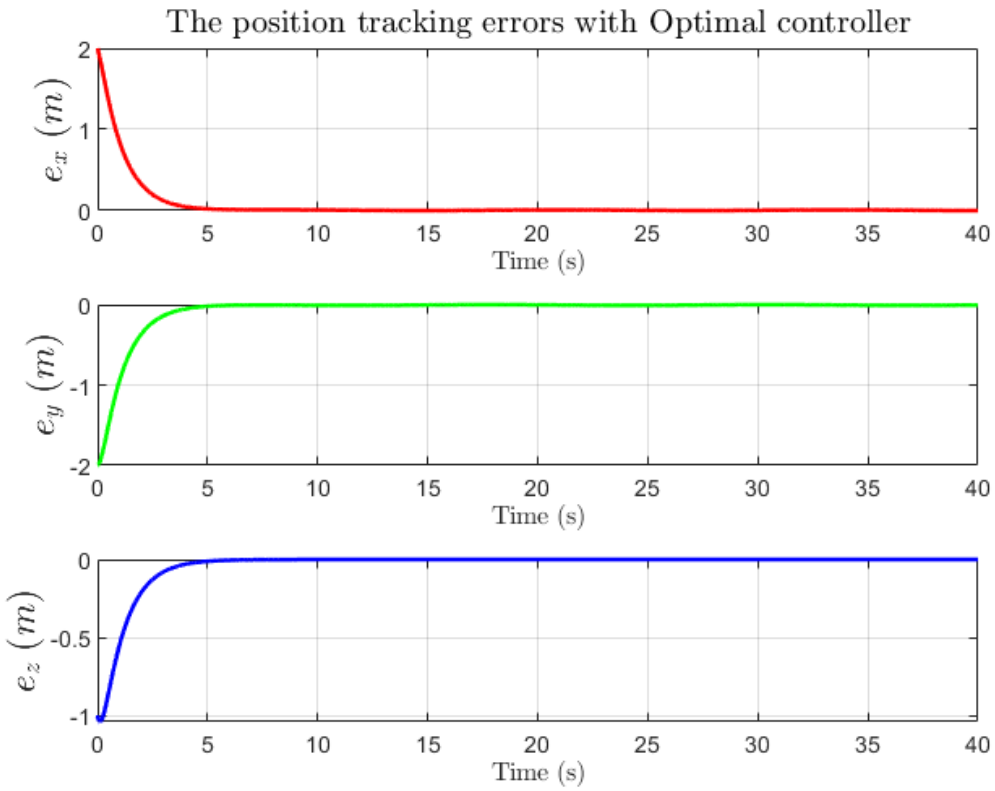


Figure 3.7: Position tracking error with Optimal controller

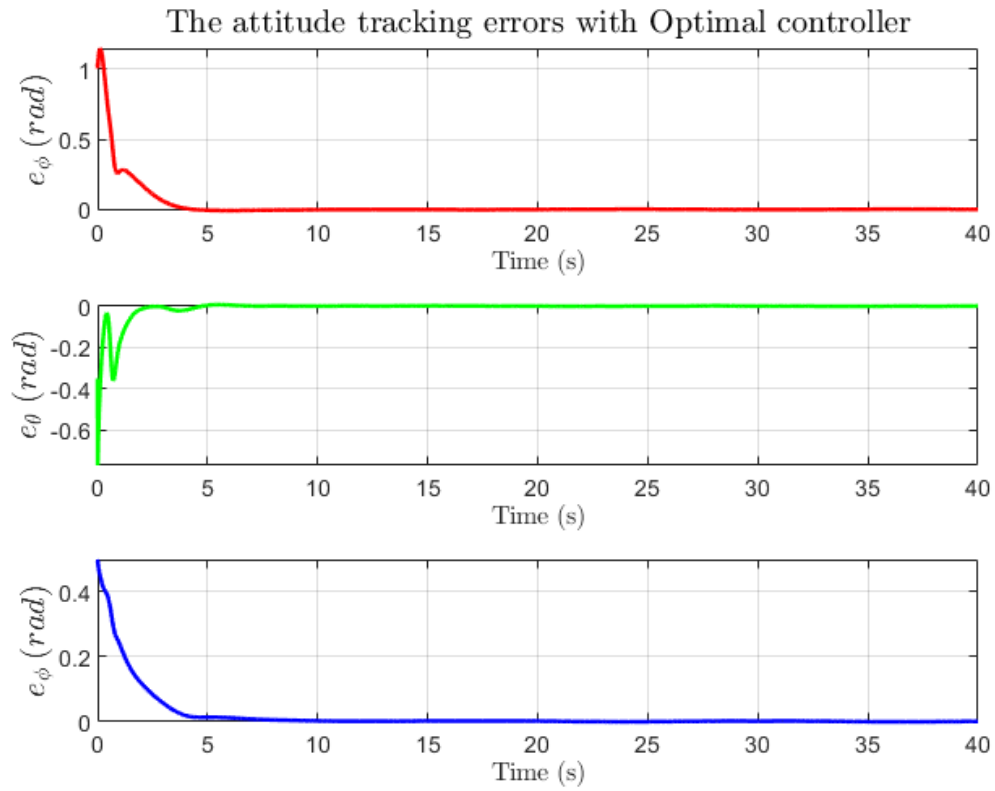


Figure 3.8: Attitude tracking error with Optimal controller

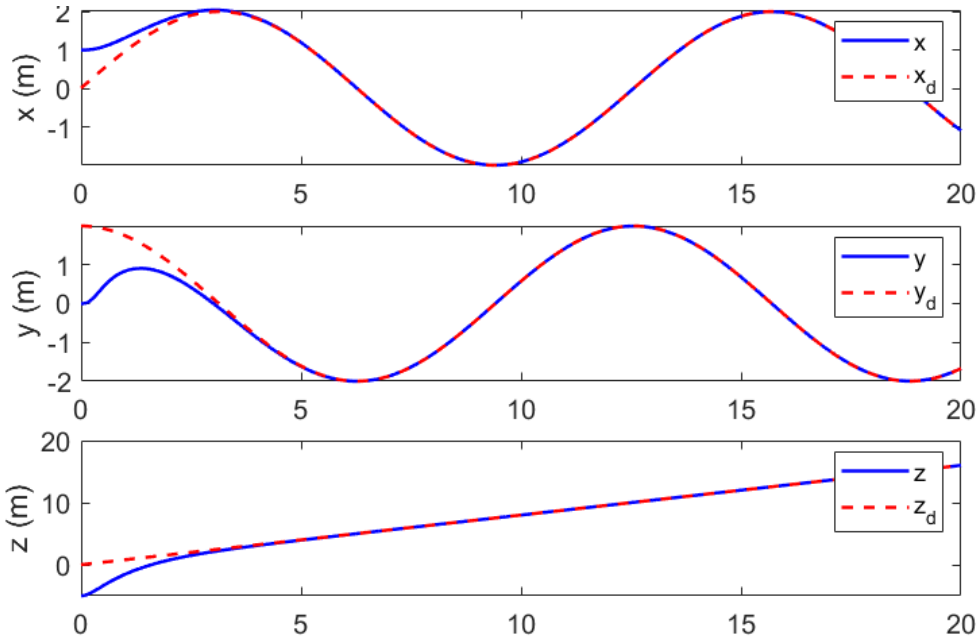


Figure 3.9: Trajectory position tracking with Optimal control

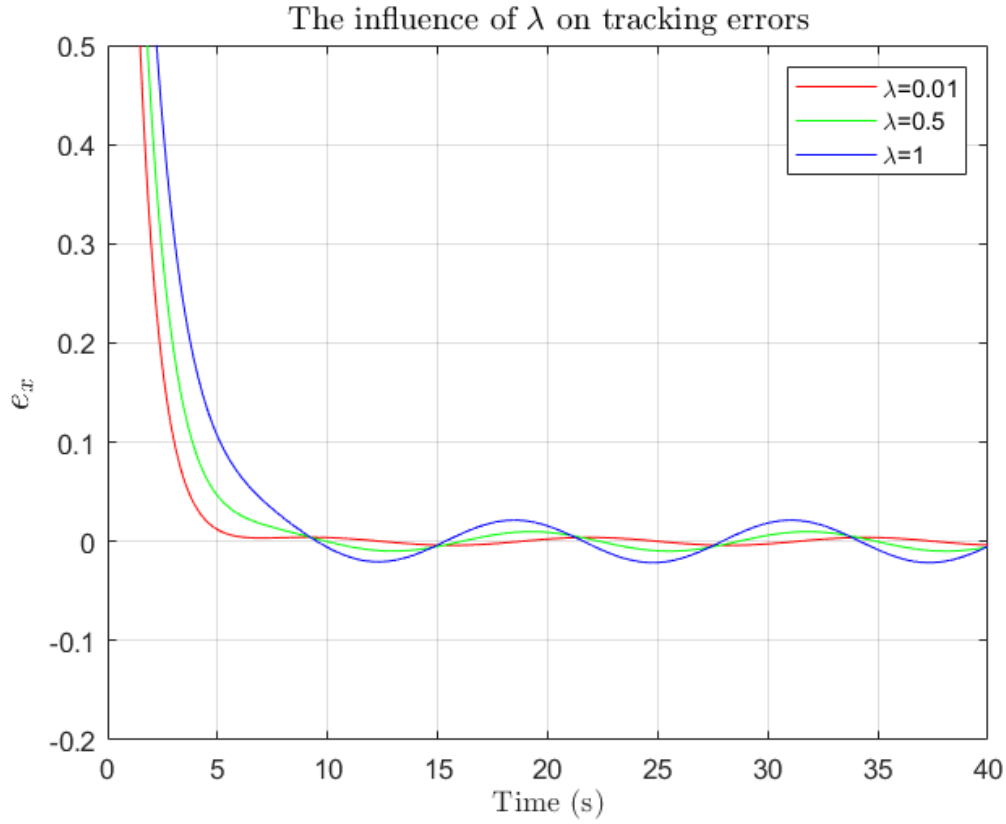


Figure 3.10: The effect of λ on control quality

From the figure (3.10), we can see that as λ increases, the tracking error increases in the sense of oscillating with larger amplitude, this is consistent with the theoretical analysis in chapter 2.

Conclusion

Through the project, we can see the effectiveness of the Data-driven PI algorithm, using data to solve the problem of optimal tracking control for linear and nonlinear systems without knowing the model. The convergence of the optimal solution and the stability of the system are proven through theorems in a strict and clear way (detailed proof in the appendix). Simulations for the quadrotor show the feasibility of applying the algorithm to real objects. However, the presented algorithm is still limited when it does not consider the influence of noise and input limit constraints of the system. In parallel, the development of swarm systems has been solving many complex control problems with high applicability. Therefore, the future development direction of the project is to consider the influence of noise and input limit constraints, solve the problem with the HJI equation, combine formation control and optimal control for multiple quadrotors systems to achieve greater applicability in practice.

Bibliography

[1] *Bellman's Principle of Optimality and its Generalizations*, pages 135–161. Springer US, Boston, MA, 2002.

[2] *REINFORCEMENT LEARNING AND OPTIMAL ADAPTIVE CONTROL*, chapter 11, pages 461–517. John Wiley and Sons, Ltd, 2012.

[3] E. Altug, J.P. Ostrowski, and R. Mahony. Control of a quadrotor helicopter using visual feedback. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 72–77 vol.1, 2002.

[4] Abhijit Das, Frank Lewis, and Kamesh Subbarao. Backstepping approach for controlling a quadrotor using lagrange form dynamics. *Journal of Intelligent and Robotic Systems*, 56:127–151, 09 2009.

[5] Bruce Finlayson and L.E. Scriven. The method of weighted residuals - a review. *Appl. Mech. Rev.*, 19:735–748, 01 1966.

[6] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. 08 2007.

[7] Rushikesh Kamalapurkar, Huyen Dinh, Shubhendu Bhasin, and Warren E. Dixon. Approximate optimal trajectory tracking for continuous-time nonlinear systems. *Automatica*, 51:40–48, 2015.

[8] Bahare Kiumarsi and Frank L. Lewis. Actor–critic-based optimal tracking for partially unknown nonlinear discrete-time systems. *IEEE Transactions on Neural Networks and Learning Systems*, 26(1):140–151, 2015.

[9] Frank Lewis and Draguna Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits and Systems Magazine, IEEE*, 9:32 – 50, 01 2009.

[10] Yibo Li and Shuxi Song. A survey of control algorithms for quadrotor unmanned helicopter. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*, pages 365–369, 2012.

[11] D. Liu, Xiong Yang, and H. Li. Adaptive optimal control for a class of continuous-time affine nonlinear systems with unknown internal dynamics. *Neural Computing and Applications*, 23:1843–1850, 2012.

[12] Ashfaq Mian and Wang Daobo. Modeling and backstepping-based nonlinear control strategy for a 6 dof quadrotor helicopter. *Chinese Journal of Aeronautics - CHIN J AERONAUT*, 21:261–268, 06 2008.

[13] Hamidreza Modares and Frank Lewis. Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica*, 50, 07 2014.

[14] Hamidreza Modares and Frank L. Lewis. Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning. *IEEE Transactions on Automatic Control*, 59(11):3051–3056, 2014.

[15] Chaoxu Mu, Changyin Sun, and Wei Xu. Fast sliding mode control on air-breathing hypersonic vehicles with transient response analysis. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 230, 11 2015.

[16] Nguyễn Doãn Phước. *Tối ưu hóa và điều khiển tối ưu*. NXB Bách Khoa Hà Nội, 2010.

[17] Richard Sutton and AG Barto. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11:126–134, 01 1999.

[18] A. Tayebi and S. McGilvray. Attitude stabilization of a vtol quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 14(3):562–571, 2006.

[19] Wei Wang, Hao Ma, and C.-Y Sun. Control system design for multirotor mav. *Journal of Theoretical and Applied Mechanics*, 51:1027–1038, 01 2013.

[20] Haojian Xu, Petros Ioannou, and Majdedin Mirmirani. Adaptive sliding mode control design for a hypersonic flight vehicle. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, 27:829–838, 09 2004.

Appendix 1. ***Prove theorem 2.2.3***

From (2.20) and (2.21) we clearly see that $V^{i+1}(X)$ *and* $u^{i+1}(X)$ *are solution of (2.21), so to prove the above theorem we only need to prove the unique existence of the solution to the equation (2.21).2.21. According to (2.20) and (2.21) we have:*

$$\begin{aligned}\frac{V^{i+1}(X)}{dt} &= \lim_{\Delta t\to 0}\frac{1}{\Delta t}(V^{i+1}(X(t+\Delta t)) - V^{i+1}(X(t))) \\ &= 2\lim_{\Delta t\to 0}\frac{1}{\Delta t}\int_t^{t+\Delta t}[u^{i+1}(X(\tau))]^T R[u^i(X(\tau)) - u(\tau)]d\tau \\ &\quad + \lim_{\Delta t\to 0}\frac{1}{\Delta t}\int_t^{t+\Delta t}\lambda * V^{i+1}(X(\tau))d\tau \\ &\quad - \lim_{\Delta t\to 0}\frac{1}{\Delta t}\int_t^{t+\Delta t}[X^T(\tau)QX(\tau) + u^T(\tau)Ru(\tau)]d\tau \\ &= 2[u^{i+1}(X(t))]^T R[u^i(X(t)) - u(t)] + \lambda V^{i+1}(X(t)) \\ &\quad - X^T(t)QX(t) - [u^i(X(t))]^T Ru^i(X(t))\end{aligned} \tag{21}$$

Suppose that there exists another solution $(S(X), v(X))$ *of the above equation. Here* $S(X) \in \mathcal{V}(\mathcal{X})$ *and* $S(X) \geq 0$, $S(0) = 0$, $v(X) \in$ *mathcalU*$(\mathcal{X})$. *Clearly* $(S(X), v(X))$ *satisfies:*

$$\begin{aligned}\frac{dS(X)}{dt} &= 2[v(X(t))]^T R[u^i(X(t)) - u(t)] + \lambda S(X(t)) \\ &\quad - X^T(t)QX(t) - [u^i(X(t))^T Ru^i(X(t))\end{aligned} \tag{22}$$

Subtracting the side from the another side of (21) gives (22) we get:

$$\begin{aligned}&\frac{d}{dt}(V^{i+1}(X) - S(X)) - \lambda(V^{i+1}(X(t)) - s(X(t))) \\ &\quad = 2[u^{i+1}(X(t)) - v(X(t))]^T R[u^i(X(t)) - u(t)]\end{aligned} \tag{23}$$

Multiplying both sides of the above equation by $e^{-\lambda t}$ *we get:*

$$\begin{aligned}&e^{-\lambda t}\frac{d}{dt}(V^{i+1}(X) - S(X)) - \lambda(V^{i+1}(X(t)) - s(X(t))) \\ &\quad = \frac{d}{dt}(e^{-\lambda t}(V^{i+1}(X)) - S(X)) \\ &\quad = 2e^{-\lambda t}[u^{i+1}(X(t)) - v(X(t))]^T R[u^i(X(t)) - u(t)]\end{aligned} \tag{24}$$

It can be seen that (24) holds $\forall u(t) \in \mathbb{R}^m$. *When* $u(t) = u^i(X(t))$, *we get:*

$$\frac{d}{dt}(e^{-\lambda t}(V^{i+1}(X)) - S(X)) = 0 \tag{25}$$

So:

$$e^{-\lambda t}(V^{i+1}(X)) - S(X) = C \tag{26}$$

Here C *is a real constant. As mentioned* $S(0) = 0$, *so* $C = e^{-\lambda t}(V^{i+1}(0)) - S(0)$, *so* $V^{i+1}(X) = S(X)$. *Substituting the above result into (24) we see:*

$2e^{-\lambda t}[u^{i+1}(X(t)) - v(X(t))]^T R[u^i(X(t)) - u(t)]$ *is true for all* $u(t) \in \mathbb{R}^m$. *Since* R *is a positive definite matrix and* $u^{i+1}(X(t)) - u(t)$ *will not be 0 all the time therefore we have* $u^{i+1}(X) = v(X(t))$.

Appendix 2. ***Prove theorem 3.2.4***

The expanded system:

$$\dot{X} = F(X) + G(X)\hat{u}^i(X) \tag{27}$$

Differentiating the solution of the Policy Evaluation equation, we get:

$$\begin{aligned}
\dot{V}^i(X) &= (\nabla V^i)^T(F + G\hat{u}^i) \\
&= (\nabla V^i)^T(F + G(u^i + \epsilon_u^i)) + X^T QX \\
&+ (u^{i-1} - u^i)^T R(u^{i-1} - u^i) - \lambda V^i - X^T QX \\
&- (u^{i-1} - u^i)^T R(u^{i-1} - u^i) + \lambda V^i \\
&= (\nabla V^i)^T F - 2(u^i)^(T)Ru^i \\
&+ (\nabla V^i)^T G\epsilon_u^i + X^T QX + (u^{i-1})^T Ru^{i-1} \\
&- 2(u^i)^T Ru^{i-1} + (u^i)^T Ru^i - \lambda V^i - X^T QX \\
&- (u^{i-1} - u^i)^T R(u^{i-1} - u^i) + \lambda V^i \\
&= (\nabla V^i)^T(F + G\hat{u}^{i-1}) + X^T QX + (u^{i-1})^T Ru^{i-1} \\
&- \lambda V^i + (\nabla V^i)^T G\epsilon_u^i - (u^i)^T Ru^i \\
&X^T QX - (u^{i-1} - u^i)^T R(u^{i-1} - u^i) + \lambda V^i \\
&= (\nabla V^i)^T G\epsilon_u^i - (u^i)^T Ru^i - X^T QX \\
&- (u^{i-1} - u^i)^T R(u^{i-1} - u^i) + \lambda V^i
\end{aligned} \tag{28}$$

Obviously the extended system will be asymptotically stable if the following condition is satisfied

$$(u^i)^T Ru^i + X^T QX + (u^{i-1} - u^i)^T R(u^{i-1} - u^i) \geq (\nabla V^i)^T G\epsilon_u^i + \lambda V^i \tag{29}$$

Since ϵ_u^i *and* λ *both approach 0, the above condition becomes*

$$(u^i)^T Ru^i + X^T QX + (u^{i-1} - u^i)^T R(u^{i-1} - u^i) \geq 0 \tag{30}$$

Therefore, $\dot{V}^i(X) \leq 0$, *so the closed system is asymptotically stable* $\rightarrow$ *the tracking error is asymptotically stable.*

CONSEQUENCE 1: Considering the above extended system with an approximate control signal, if neither ϵ_u^i *nor* λ *is equal to* 0, *the tracking error can be arbitrarily small by choose a suitable set of activation functions, the coefficient* λ *is small enough and the penalty matrix* Q_e *is large enough.*

PROVE THE CONSEQUENCES:

The first component in the right-hand side of condition (5.3) depends mainly on the approximate error ϵ_u^i, *which can be arbitrarily small by choosing an*

appropriate set of activation functions based on the properties monotonic approximation of a neural network, e.g. $||\epsilon_u^i|| \leq \epsilon_{uM}$, *where* ϵ_{uM} *is a positive constant.*
The second component is influenced by both λ *and the function* V^i. *We divide into the following 2 cases*
- In case $\lambda = 0$, *it means that the control signal also converges to 0 when the system state converges to 0, condition (5.3) becomes*

$$(u^i)^T R u^i + X^T Q X + (u^{i-1} - u^i)^T R (u^{i-1} - u^i) \geq \nabla V_M G_M \epsilon_{uM} \tag{31}$$

The above condition can be easily satisfied by choosing a suitable set of activation functions to make ϵ_{uM} *small enough and achieve the bound condition of tracking error. - In case* $\lambda \neq 0$, *condition (5.2) becomes:*

$$(u^i)^T R u^i + X^T Q X + (u^{i-1} - u^i)^T R (u^{i-1} - u^i) \geq \nabla V_M G_M \epsilon_{uM} + \lambda V_M \tag{32}$$

From there, the coefficient λ *can be chosen small enough to satisfy the bound condition of the tracking error*
Furthermore, multiplying both sides of (5.2) by $e^{\lambda t}$, *we get*

$$\frac{\partial}{\partial}(e^{-\lambda t} V^i(X)) = e^{-\lambda t}((\nabla V^i)^T G \epsilon_u^i - (u^i)^T R u^i - X^T Q X - (u^{i-1} - u^i)^T R (u^{i-1} - u^i)) \tag{33}$$

The derivative component on the left side of the above condition is negative for $Q_e > 0$ *and condition (36) is satisfied, so the tracking error will decrease until the component* $e^{-\lambda t}$ *equals 0 or condition (36) is not satisfied. In fact, choosing* Q_e *larger will increase the rate of reduction of the derivative component to zero and the tracking error will be smaller. This also means that* $V^i(X)$ *does not reach the bound on* V_M *in condition (37). Therefore, choosing a larger* Q_e *also makes the tracking error smaller.*

Appendix 3. *Điều kiện chặn trên của* λ

We consider the system $\dot{X} = F(X) + G(X)u$ *in formula (2.9)-(2.11) in partially linearized form, rewritten as* $\dot{X} = AX + Bu + \tilde{F}(X)$. *With* $\tilde{F}(X)$ *being the component after partial linearization to ensure the system is still nonlinear.*
Here $A = [A_1, A_1 - A_2; 0, A_2]$, $B = [B_1; 0]$, A_1, A_2 *are linearized matrices of* $f(x)$ *and* $r_d(x_d)$.
We can state in the form of a theorem that: For the system $\dot{X} = F(X) + G(X)u$ *to have an optimal solution to make the system error asymptotically stable, the above bound condition of* λ *must satisfy:*

$$\lambda \leq \bar{\lambda} = 2\|(B_1 R^{-1} B_1^T Q_e)^{1/2}\| \tag{34}$$

Proof:

From [2], the Hamilton function is defined as follows:

$$\begin{aligned} H^a &= e^{\lambda t} H^b(\mu, u^*) \\ &= e^{\lambda t}(e^{-\lambda t}(X^T Q X + u^{*T} T u*) + \mu^T(F + G u^*)) \\ &= X^T Q X_u^{*T} R u^* + \nu^T(F + G u^*) \end{aligned} \tag{35}$$

The optimal solution u^* simultaneously satisfies the following two states:

$$\dot{X} = H_\nu^a(X, \nu) \tag{36}$$

$$\dot{\nu} = \lambda \nu - H_X^a(X, \nu) \tag{37}$$

Here, μ is a co-state variable, $\mu = e^{\lambda t}\mu$, $H_\mu^a = \partial H_\mu^a / \partial \mu$ and $H_X^a = \partial H_X^a / \partial X$ From there (35) is transformed into:

$$H^a = X^T Q X + u^{*T} R u^* + \nu^T(AX + Bu^* + \tilde{F}(X)) \tag{38}$$

On the other hand, the solution of the equation (37) is equivalent to the solution of the equation HJB while $\nu = \nabla V^*$, then ν can be described as:

$$\nu = 2PX + f_a(X) \equiv \bar{\nu} + f_a(X) \tag{39}$$

And the optimal solution:

$$u^* = -R^{-1} B^T P X + f_b(X) \tag{40}$$

Here $f_a(X)$ is the nonlinear part and $f_b(X)$ depends on $f_a(X)$, $\tilde{F}(X)$ and P. Definition $P = [P_{11}, P_{12}; P_{21}, P_{22}]$ and using (38)-(40) the kinematic tracking error can be written as:

$$\dot{e}_d = (A_1 - B_1 R^- 1 B_1^T P_{11}) e_d + \tilde{F}_a = A_m e_d + \tilde{F}_a \tag{41}$$

Therefore we have the ARE equation below:

$$Q_e + A_m^T P_{11} + P_{11} A_m - \lambda P_{11} + P_{11} B_1 R^{-1} B_1^T P_{11} = 0 \tag{42}$$

Multiplying both sides of the above equation by e_d^T on the left side and e_d on the right side we get:

$$2(Re(\rho) - 0.5\lambda) e_d^T P_{11} e_d = -e_d^T Q_e e_d - e_d^T P_{11} B_1 R^{-1} B_1^T P_{11} e_d \tag{43}$$

With ρ being the partial solution of A_m, since $P_{11} > 0$ we get:

$$(Re(\rho) - 0.5\lambda) \leq -\|(Q_e P_1 1^{-1})^{1/2}\| \|(P_{11} + P_{11} B_1 R^{-1} B_1^T P_{11})^{1/2}\| \tag{44}$$

Therefore: $Re(\rho) \leq -\|(B_1 R^{-1} B_1^T Q_e)^{1/2}\| + 0.5\lambda$ (proved.)