Online reinforcement learning control for aerospace systems

Zhou, Ye

DOI
10.4233/uuid:5b875915-2518-4ec8-a1a0-07ad057edab4

Publication date
2018

Document Version
Final published version

Citation (APA)
Zhou, Y. (2018). *Online reinforcement learning control for aerospace systems*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:5b875915-2518-4ec8-a1a0-07ad057edab4

Online Reinforcement Learning Control for Aerospace Systems

Online Reinforcement Learning Control for Aerospace Systems

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 11 april 2018 om 15:00 uur

door

Ye ZHOU

ingenieur luchtvaart en ruimtevaart
geboren te Hefei, Anhui, China

Dit proefschrift is goedgekeurd door de promotoren:

prof. dr. ir. M. Mulder en dr. Q. P. Chu

Copromotor:

dr. ir. E. van Kampen

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. M. Mulder,	Technische Universiteit Delft, promotor
Dr. Q. P. Chu,	Technische Universiteit Delft, promotor
Dr. ir. E. van Kampen,	Technische Universiteit Delft, copromotor
Onafhankelijke leden:	
Prof. dr. J. Si,	Arizona State University
Prof. dr.-ing F. Holzapfel,	Technische Universität München
Prof. dr. D. G. Simons,	Technische Universiteit Delft
Prof. dr. R. Babuska,	Technische Universiteit Delft

Keywords: Reinforcement Learning; Aerospace Systems; Optimal Adaptive Control; Approximate Dynamic Programming; Adaptive Critic Designs; Incremental Model; Nonlinear Systems; Partial Observability; Hierarchical Reinforcement Learning; Hybrid Methods.

Printed by: Ipskamp Printing.

Front & Back: Designed by Ye Zhou.

ISBN 978-94-6366-021-1

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

To my beloved parents, husband, and little daughter . . .

SUMMARY

ONLINE REINFORCEMENT LEARNING CONTROL FOR AEROSPACE SYSTEMS

Ye ZHOU

Recent technological improvements have spurred the development of innovative and more advanced aerospace systems. The increased complexity of these systems has become one of the major challenges of the aerospace control system design. The multi-objective tasks in various applications, ranging from the air domain to space domain and from military use to commercial use, also increase the automatic control requirements and complexity. Furthermore, the uncertainties in aerospace systems, such as changing shapes of morphing aircraft, and in the environment, such as sudden gusts, complex air traffic, and space debris impacts, have also heightened the need for online adaptability in control systems. To meet the growing complexity of the system dynamics, the increasing difficulty of control tasks, and the demanding requirement of adaptability, aerospace systems are in urgent need of higher levels of autonomy.

The complexity and diversity of aerospace systems and autonomous control tasks motivate researchers to explore intelligent methods. Intelligent autonomous aerospace systems, on the one hand, need to learn the current system dynamics and the environment online and control the system adaptively and accurately. On the other hand, these systems also need to be able to trade off among multiple objectives and retain safety. Therefore, a complete intelligent system often has a hierarchical control architecture. The low-level control ability is the foundation of the higher levels and limits the improvement of the whole autonomous control system. This limitation is one of the main reasons for the fact that many existing high-level autonomous algorithms cannot be successfully applied yet to real aerospace systems. Besides, the intelligence and autonomy of high-level decision-making systems are also in need of improvement, to meet the new challenges in current and future aerospace systems, such as deep-space exploration, indoor guidance and navigation, and self-organized swarm formation.

Reinforcement Learning (RL) is a framework of intelligent, self-learning methods that can be applied to different levels of autonomous operations and applications. It links bio-inspired artificial intelligence techniques to the field of control and decision-making. RL methods, in the low-level control field, can be used to improve the control efficiency and adaptability when the dynamical models are unknown or uncertain.

These control problems, such as stabilization and reference tracking, are often modeled in continuous state and action spaces. RL methods, in the high-level decision-making field, can be applied to enhance the intelligence of planning and to ensure the coordination with the low-level control. In these problems, state and action spaces can be discrete, continuous or even hybrid.

RL methods are relatively new in the field of aerospace guidance, navigation, and control. They have many benefits, but also some limitations, when applied to aerospace systems. This dissertation aims to deal with the following main research question:

> **How can aerospace systems exploit RL methods to improve the autonomy and online learning with respect to the a priori unknown system and environment, dynamical uncertainties, and partial observability?**

This main research question is addressed in three parts, for three specific RL methods and applications: (i) Approximate Dynamic Programming (ADP) for control problems with an approximately convex true cost-to-go, (ii) Adaptive Critic Designs (ACDs) for general nonlinear control problems, and (iii) Hierarchical Reinforcement Learning (HRL) for high-level guidance and navigation. This leads to the following research questions:

1. How to generalize Linear Approximate Dynamic Programming (LADP) to deal with nonlinear and/or time-varying systems, model mismatch, and partial observations, while retaining the efficiency and mathematical explicitness?

2. How to devise online adaptive critic designs and improve the online adaptability, to cope with internal uncertainties, external disturbances, and even sudden faults?

3. How to establish a systematic hierarchical reinforcement learning controller that deals with multiple objectives and partial observability, possesses transfer learning ability, and utilizes diverse RL methods?

To address the first question, this dissertation proposes *incremental* Approximate Dynamic Programming (iADP) methods. Instead of using nonlinear function approximators to approximate the true cost-to-go, iADP methods use an (extended) incremental model to deal with the nonlinearity of unknown systems and uncertainties of the environment. These methods can still apply a quadratic cost function to generate an efficient and mathematically explicit optimal control algorithm. These methods do not need any a priori knowledge of the system dynamics, online identification of the global model, nor even an assumption of the time scale separation, but only an online identified (extended) incremental model.

The iADP method is first proposed to solve regulation problems for nonlinear systems. When the direct measurement of the full state is available, the incremental model can be identified to predict the next state. With this prediction and a quadratic cost function, the control increment can be calculated adhering to the optimality principle. When the only measurements are the input/output of the dynamical system, the optimal control increment is calculated with an output feedback algorithm and an extended incremental model. This method is applied to an unknown nonlinear missile model, with both full state and output measurements, to iteratively optimize the flight control policy. The simulation results demonstrate that the iADP method improves the closed-loop performance of the nonlinear system, while keeping the design process simple and systematic.

The concept of iADP is further expanded to tracking problems for Multiple-Input Multiple-Output (MIMO) nonlinear systems and to partial observable control problems. Because iADP methods have a separate structure to represent the local system dynamics, the cost function can be less dependent on the system or the reference, and only needs to be a rough approximation of the cost-to-go. This approximation is a quadratic function only of the current tracking error, without expanding the dimension of the state space for the cost function to an augmented one.

Two observability conditions are considered in this tracking control problem. When the direct measurement of the full state is available, the incremental model can be online identified to design the optimal control increment. In addition, when the only measurement is the output tracking error, involved with tracking a stochastic dynamical reference, the system becomes partially observable. The observations are used to identify the extended incremental model and to predict the next output tracking error for the optimal tracking control. For each observability condition, an off-line learning algorithm is applied to improve the policy iteratively until it is accurate enough, and hereafter an online algorithm is applied to update the policy recursively at each time step. The recursive algorithms can also be used online in real systems which may be different from the system model used in the iterative learning stage. These algorithms are applied to an attitude control problem of a simulated satellite disturbed by liquid sloshing. The results demonstrate that the proposed algorithms accurately and adaptively deal with time-varying internal dynamics, while retaining efficient control, especially for unknown nonlinear systems with only partial observability.

To answer the second research question, this dissertation develops online ACDs based on the *incremental model*. ACDs can generally be categorized into three groups: 1) Heuristic Dynamic Programming (HDP), 2) Dual Heuristic Programming (DHP), and 3) Globalized Dual Heuristic Programming (GDHP). Besides, action dependent variations of these three original versions have been developed by directly connecting the output of the actor to the input of the critic. This dissertation focuses on action independent ACDs, specifically HDP and DHP.

An *Incremental model* based Heuristic Dynamic Programming (IHDP) method is proposed to online and adaptively control unknown aerospace systems. This method replaces the global system model approximator with an incremental model. This approach, therefore, does not need off-line training stages and may accelerate online learning. The IHDP method is compared with conventional HDP in an online tracking control of the unknown nonlinear missile model. The results show that the presented IHDP method speeds up the online learning, has a higher tracking precision, and can deal with a wider range of initial states than the conventional HDP method. In addition, the IHDP method is also applied to the MIMO satellite attitude tracking control disturbed by liquid sloshing and with sudden external disturbances. The simulation results also demonstrate that the IHDP method is adaptive and robust to internal uncertainties and external disturbances.

To further improve the control performance and accelerate the online learning, an *Incremental model* based Dual Heuristic Programming (IDHP) method is developed. The IDHP method uses a Recursive Least Square (RLS) approach, to identify in real-time the incremental model instead of the global system model. In addition to the online reference tracking problem, a Fault-Tolerant Control (FTC) task is performed using IDHP and conventional DHP. The results demonstrate that the IDHP method can successfully control a faulty and unstable system adaptively before the states diverge, where DHP fails. To further validate the robustness of the proposed IDHP method, high-frequency measurement noise is superimposed to the measurements of system states. The simulation results indicate that the IDHP method is not sensitive to the measurement noise.

The third research question is answered through the development of the *hybrid* Hierarchical Reinforcement Learning (hHRL) method, for guidance and navigation problems. This method consists of several hierarchical levels, where each level uses different methods to optimize the learning with different types of information and objectives. The explicit rules of establishing the hierarchies, decomposing the tasks, and assigning the rewards are formulated. Detailed implementations of the proposed hHRL method are presented for an online, multi-objective guidance and navigation task with partial observability and multiple objectives (i.e., approaching a target area while avoiding obstacles).

The proposed method is first applied to a benchmark maze, to prevent collision online and to improve the performance of approaching the target episodically. The result is compared to a ‘flat’ RL method and a single-method HRL method and indicates that the proposed hHRL method is more efficient in dealing with the ‘curse of dimensionality’ and in reducing the uncertainty or ambiguity. The learned results are then applied to a different, expanded maze, which validates that learning results can indeed be transferred across tasks to speed up learning in new tasks or environments. Lastly, the same method is applied to a non-stationary environment with modified sensors and a partial map. The hHRL method, using relative micro states and absolute macro states in different hierarchical levels, allows for learning in non-stationary environments without loss of efficiency. These results indicate that the proposed hHRL method can help to accelerate learning, to alleviate the ‘curse of dimensionality’ in complex decision-making tasks, to reduce the uncertainty or ambiguity, to transfer the learned results within and across tasks efficiently, and to be applied to non-stationary environments. This proposed method can potentially design a near-optimal policy hierarchically for autonomous guidance and navigation with an unknown system and environment.

In conclusion, this dissertation contributes with several methods that improve the intelligence and autonomy of aerospace systems. These improvements are mainly from three perspectives: 1) enhancing the adaptability and efficiency of low-level control, 2) improving the intelligence and online learning ability of guidance, navigation, and control, and 3) creating a well-organized hierarchy to ensure coordination between each level. The proposed methods provide novel insights for both the reinforcement learning research community and for developers of aerospace automatic control system.

Contents

1

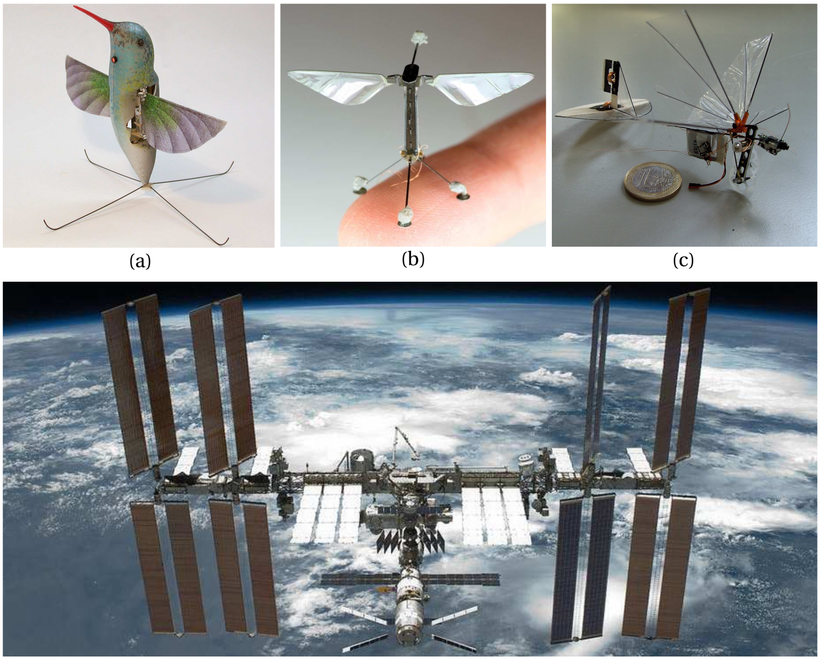
Introduction

1.1. Autonomous Control in Aerospace Systems

The last few decades have seen rapid advances of automated control in many domains, ranging from industrial manufacturing and household appliances to unmanned aerial vehicles. The emergence of self-driving cars on the road and household robots at home brings advanced automated control into our everyday lives. However, most of these systems can only complete their tasks under similar, predictable circumstances for which they are designed. Autonomous control systems, on the other hand, must be able to change their behaviour to unexpected situations in both the system and environment [1, 2]. Aerospace systems are urgently in need of higher levels of autonomy, to meet the growing complexity of the dynamical systems, the increasing difficulty of control tasks and unmanned operations, and the demanding requirement of adaptability.

Recent technological improvements have spurred the development of innovative aerospace systems. Since then, system complexity has become one of the major challenges of control design for these aerospace systems. The bio-inspired ornithopters, such as Nano-Hummingbird (by AeroVironment) [3], RoboBee (by Harvard University) [4, 5], and DelFly (by Delft University of Technology) [6, 7], in Fig. 1.1(a)-(c), are examples of extremely complex aerial vehicles. First, the aerodynamics and kinematics of the flapping wings interact with each other, which impedes the analysis of the force and moment mechanisms. Second, the high nonlinearity complicates the identification of the system dynamics. Thus, it is almost impossible to build globally accurate mathematical models for these systems [8, 9]. Other examples are increasingly sophisticated spacecraft, involving complex internal dynamics, as presented in Fig. 1.1(d). Here, liquid sloshing is one of the unknown and uncertain internal dynamics interacting with the motion of the vehicle [10–12]. Although it has been studied for many years, an accurate liquid sloshing model is extremely difficult to obtain [12, 13].

With the rapid development and evolution, aerospace systems have already had a multitude of applications ranging from the air domain to space domain. Air domain



(a) (b) (c)

(d)

Figure 1.1: Examples of complex aerospace systems. Aerial vehicles: (a) Nano-Hummingbird[3], (b) RoboBee[5], (c) DelFly Micro[7]; Space vehicle: (d) International Space Station[14].

applications include reconnaissance, surveillance, and targeting for military use, transportation for civilian use, aerial photography for commercial use, etc. Spacecraft can be used for space exploration, obtaining observations from a different perspective on Earth phenomena, and for the multi-domain task, launching communication satellites and space telescopes. Many systems are demanded to execute multiple tasks, while retaining safety and performance, without direct human supervision. The growing control requirements and multiple objectives also increase the control complexity.

Furthermore, there are growing demands for adaptability that allows the control of unknown and/or time-varying systems in the presence of uncertainties. Compared to conventional aircraft, new types of aerospace systems are far more complex and uncertain, e.g., convertiplane with large angle maneuvers [15] and morphing aircraft with changing shapes [16, 17]. Besides uncertainties in the system, the uncertainties in the environment, such as sudden gusts, complex air traffic, and space debris impacts, have also heightened the need for online adaptability in control systems.

The complexity and diversity of aerospace systems and control tasks motivate researchers to explore autonomous control methods. Intelligent autonomous aerospace systems, on the one hand, need to learn the current system dynamics and the environment online and control the system adaptively and accurately. On the other hand, these systems need to trade off among multiple objectives and retain safety. Therefore, a complete intelligent system often has a hierarchical control architecture, where the higher intelligence requires the lower precision [18, 19]. The intelligence and autonomy of an aerospace system can be increased from three aspects: 1) enhancing the adaptability and efficiency of low-level control, 2) improving the intelligence and online learning ability of guidance, navigation, and control, and 3) creating a well-organized hierarchy to ensure coordination between each level. The low-level control ability is the foundation of the higher levels and limits the improvement of the whole autonomous control system. This limitation is one of the main reasons for the fact that many existing high-level autonomous algorithms cannot be successfully applied yet to real aerospace systems [20–22].

Conventional control methods in aerospace are based on piecewise mathematical models of the physical system, and then generate decentralized controllers around each operating point by using appropriate methods such as stability analysis or manual tuning [1]. This design procedure relies on an accurate model and usually takes several iterations, each of them incurring a considerable cost for evaluating and testing the designed control system and also significantly extending the time to develop new models. Adaptive control, which is an active field since the 1960s, has been proposed for complex dynamical systems. Although model-based control strategies have been extensively studied and successfully applied to many applications [23–28], they are reliant on an accurate system model or on its identification. However, in many real aerospace applications, an accurate model of the complex system is often not available, nor is its identification trivial [8, 9]. In addition, when uncertain dynamics or environments are involved, the mismatch between the model and real system may degrade the control performance of model-based methods.

Besides the low-level control methods, the intelligence and autonomy of high-level decision-making systems are also in need of improvement, to meet the new challenges in current and future aerospace systems, such as deep-space exploration, indoor guidance and navigation, and self-organized swarm formation. Most of the current aerospace systems are only equipped with limited autonomy and are controlled by human pilots remotely and intensively. Some others, although mostly in academic research laboratories, can be preset to perform certain tasks, such as the RoboBee (by Harvard University) and the DelFly (by Delft University of Technology). However, in the aforementioned new applications, the systems might not have a static environment, perfect observations, or access to human control. Therefore, in the higher level, the control system needs to deal with the complexity of tasks and environment, the partial observability of the system and environment, and the transition of objectives and/or control requirements.

Reinforcement Learning (RL) is a framework of intelligent, self-learning methods that can be applied to different levels of autonomous operations and applications. This method links bio-inspired artificial intelligence techniques to the field of control and decision-making, to overcome some of the limitations and problems in traditional methods that demand precise models [29–31]. In the low-level control field, RL methods are mainly used to improve the control efficiency and adaptability when the dynamical models are unknown or uncertain [31–34]. These control problems, such as stabilization and reference tracking, are often modeled in continuous state and action spaces. In the high-level decision-making field, RL methods are applied to enhance the intelligence of planning and also to ensure the coordination with the low-level control [35–38]. In these problems, state and action spaces can be discretized, continuous or even hybrid.

1.2. Challenges in Reinforcement Learning Controllers

Reinforcement Learning methods learn to take actions that affect the system states, such as attitude angles, rotational rates, and positions, to maximize some numerical reward from interaction with the environment (Fig. 1.2). Traditional RL methods were devised for discrete state and action spaces, such as Q-learning and Sarsa, by using a lookup table [31]. As with increased real-life applications, in particular optimal control problems, RL methods have been confronted with high-dimensional, continuous spaces, which can lead to an exponential growth of states and actions known as the 'curse of dimensionality'[21, 29, 32, 34, 39].

To tackle these problems, RL methods can apply function approximators, which turn them to as Approximate Dynamic Programming (ADP) methods [22, 32, 39]. Within this category, Linear Approximate Dynamic Programming (LADP) and Adaptive Critic Designs (ACDs) have been extensively studied. Although LADP and ACDs are both continuous RL methods, they are categorized into different groups in terms of their memory structures [31, 40, 41]. LADP methods are critic-only methods, which only have state(-action) value functions and rely on the optimality principle to calculate the action. On the other hand, ACDs are actor-critic methods, which have separate memory structures to represent the policy and value function independently.

Different RL methods have their appealing benefits, successful applications as well as limitations and challenges. When applied to aerospace systems, current RL methods are often confronted with three main challenges:

- Efficiency of reinforcement learning control for nonlinear, unknown, and partially observable systems.
- Generalization and online adaptability of reinforcement learning control for unknown or faulty nonlinear systems.
- Systematic and transferable reinforcement learning methods for high-level guidance and navigation.

These will all be further explained in this section.

1.2.1. Efficiency of RL Control for Nonlinear, Unknown, and Partially Observable Systems

ADP is a RL method which applies function approximators to solve optimality problems. This function approximator can approximate the value/cost of any state in the state space. This approximate function caches information, such as value, cost, and/or the Temporary Difference (TD) error, from the observed states, and then generalizes to similar, previously unseen states. Ultimately, it can represent the utility of any state in the state space and exploit this information to achieve the overall goal.

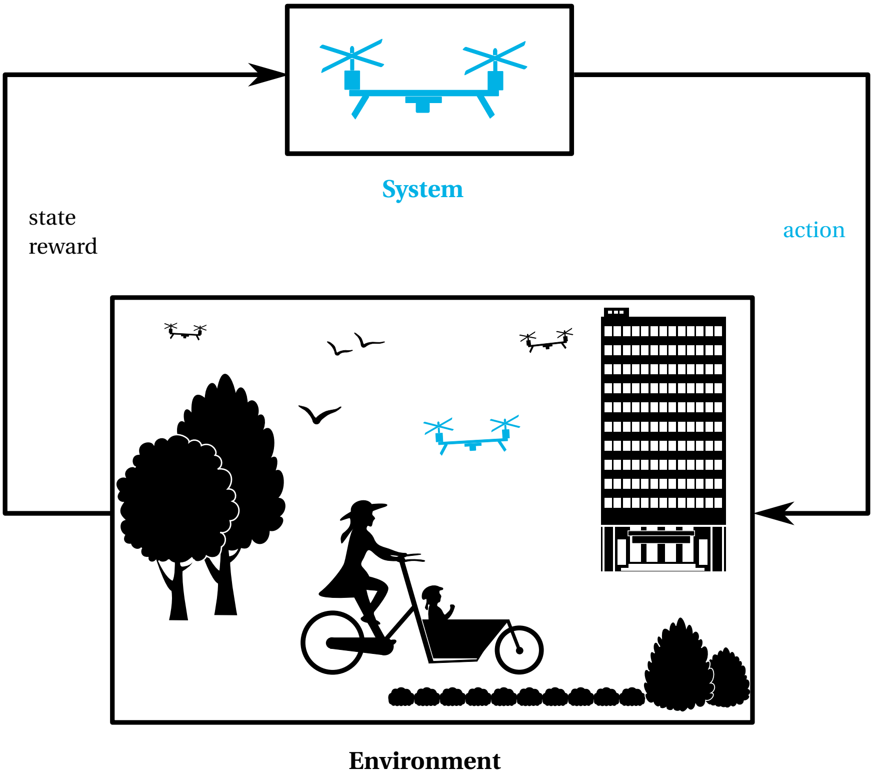


Figure 1.2: An example of the system-environment interaction with Reinforcement Learning. In this example, the system represents an air vehicle. The environment encompasses everything that surrounds and may change the system states, including the stationary obstacles, such as trees and buildings, and non-stationary obstacles, such as human beings, animals, and even other air vehicles. The system, at each moment, observes its state in this environment and may get a reward for being in that state, and then decides what actions to take to affect its state in that environment.

A widely used model-free ADP method for linear systems is the linear approximate dynamic programming method, including Value Iteration (VI) algorithms and Policy Iteration (PI) algorithms with a quadratic value function [33, 42, 43]. These methods, from a control-theoretic perspective, use a TD method to numerically approximate the kernel matrix in a Linear-Quadratic Regulator (LQR) when the system is linear but unknown, and from a RL perspective, use a quadratic cost function to approximate the true cost-to-go and to learn it adaptively. Based on this simple and efficient quadratic cost function, LADP can provide an explicit solution to linear optimal control problems [44]. Although model-free, efficient and adaptive, LADP methods were devised based on the assumption that the dynamical system is Linear Time-Invariant (LTI). All the system information is implicitly contained in the quadratic cost function. These methods, therefore, have difficulties solving problems with nonlinear or time-varying systems.

In addition to nonlinearity and complexity, real aerospace systems often involve system uncertainties and partial observability. System uncertainty includes internal uncertainties, such as the unknown aerodynamics in aerial vehicles and internal dynamics in spacecraft, and external uncertainties, such as gust and space debris impact. Most of the current LADP methods are based on an iterative off-line policy adaptation. When uncertain dynamics are involved, the model mismatch/change and the never-experienced situations, such as sudden gust or new environment, may significantly degrade the performance of off-line learned LADP methods.

Conventional RL assumes the availability of full states. However, partial observability also happens in real applications, when the system does not have enough information to infer its real states [45]. Those methods dealing with deterministic systems and measurements are often referred to as output feedback methods [33, 46, 47]. When stochastic, time-varying dynamics are involved, they belong to Partially Observable Markov Decision Processes (POMDPs) [36, 48, 49] and bring about additional challenges.

1.2.2. Generalization and Online Adaptability of RL Control for Unknown or Faulty Nonlinear Systems

Another class of ADP methods, adaptive critic designs, have shown great success in optimal adaptive control of more general nonlinear problems [32, 34, 50, 51]. They are also well known as Actor-Critic methods (ACs) because they separate evaluation (critic) and improvement (actor) using parametric structures [50]. Although they are called ACs, they often need an extra structure to approximate the global system model so as to close the update path of the actor, the critic, or both. Compared to LADP methods, ACDs can be used to control highly nonlinear systems by exploiting more complex function approximators, such as Artificial Neural Networks (ANNs) [32, 34, 50, 52].

Nevertheless, like other adaptive control methods, ACDs in one form or another still rely on off-line and/or online identification of system dynamics and adaptation of control laws. In practice, the online identification of the global system model is hard to achieve due to the unavailability of global input/output data in online tasks and the complexity of nonlinear systems [8, 9, 23–27]. Therefore, ACDs often have two learning phases [50, 51, 53, 54]: off-line learning and online learning. The off-line identification stage still needs representative simulation models, however, which are difficult to obtain.

Furthermore, during the online phase, the global model adaptation has to be sufficiently quick and smooth to cope with unforeseen dynamics, such as the resulting changes from the changes in the actor, a time-varying component in the system, uncertainties in the environment, and unexpected changes due to failures. When the global system model is approximated by function approximators, their complexity will affect the convergence speed and smoothness of the online adaptation. Several studies [55, 56] have therefore suggested to remove the global system model and to exploit previous critic outputs and/or inputs instead. Although this technique has been successfully applied to many ACD methods, it can only relieve the off-line learning phase of some Action Dependent (AD) forms. The AD variations of ACDs directly connect the output of the actor to the input of the critic [51, 53, 54, 56, 57]. From a theoretical point of view, the actor output is not necessarily an input to the critic; and from a practical perspective, the additional input can increase the dimension and complexity of the critic. Furthermore, previous studies comparing ACDs and their AD forms have reported that ACDs have higher success rates and online adaptability [51, 54]. Therefore, online learning control with ACDs is still one of the most active areas in RL today.

1.2.3. Systematic and Transferable RL Methods in High-level Guidance and Navigation

In addition to low-level control, RL methods are also widely applied to high-level guidance and navigation tasks. Traditional RL methods, which solve optimal control problems of Markov Decision Processes (MDPs) [29, 31], have been well studied for these tasks in known or small-scaled environments. However, aerospace applications can have a huge amount of states and actions, and consequently, 'curse of dimensionality'. This phenomenon is caused not only by the high dimensionality of state and action spaces but also by the complexity of the environment and task, which often impede RL applications to solve these problems. Although ADP methods can relieve this situation somewhat, the number of parameters will still grow with the exponentially growing number of states and actions, especially for complex guidance and navigation tasks with multiple objectives [58, 59].

Furthermore, in practice, aerospace systems often need to explore an initially unknown and uncertain environment with limited sensors, which is known as Partially Observable Markov Decision Processes [36, 45, 48, 49, 60]. They do not have a perfect perception of the absolute states in the environment, such as exact positions. Instead, they observe a relative state, such as heading angles and images captured by a camera. The sensed relative states can be ambiguous and prevent the value/cost of that state from converging. These problems significantly impede the application of RL methods to guidance and navigation tasks in aerospace systems.

Recent research has sought to deal with these problems through Hierarchical Reinforcement Learning (HRL) [32, 37, 61, 62]. It replaces the state-to-action mapping by a hierarchy of abstract actions. These ideas are inspired by human learning and decision making, such as illustrated in Fig.1.3. Hierarchical decomposition speeds up learning in an efficient way and naturally reduces the uncertainty induced by the partial observability. However, detailed HRL designs and their applications have barely been reported. The reason can be that the explicit rules for establishing the hierarchies still need expert knowledge, and the learned results in one application cannot be directly used in other applications. Thus, it is essential to develop a more systematic design of HRL possessing transferable learning capabilities.

1.3. Research Questions, Methods, and Scope

This section presents the research questions, the methods and contributions, and the scope and limitation of this dissertation.

1.3.1. Research Questions

RL methods are relatively new in the field of aerospace guidance, navigation, and control. They have many benefits, but also some limitations, when applied to aerospace systems. The aim of this thesis is to address the previously mentioned knowledge gaps by dealing with the following research question:

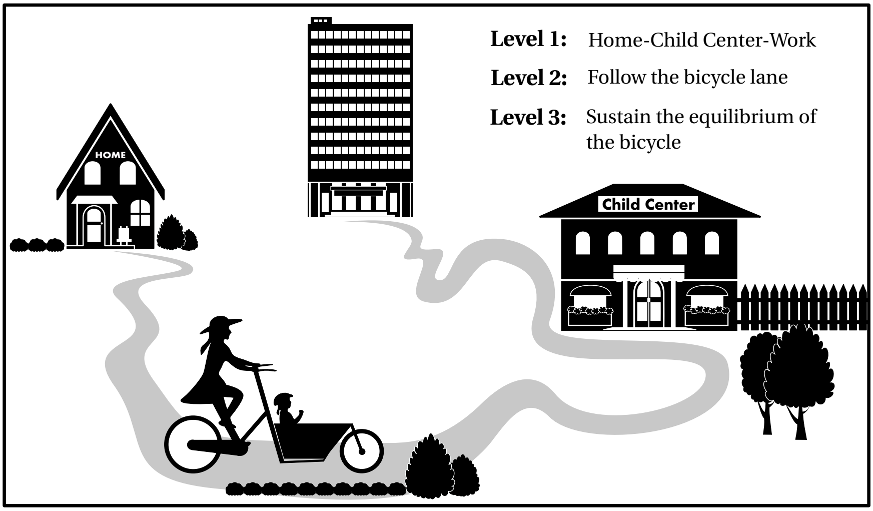

Level 1: Home-Child Center-Work
Level 2: Follow the bicycle lane
Level 3: Sustain the equilibrium of the bicycle
HOME
Child Center

Figure 1.3: HRL on human behaviors: an example of human learning and decision making with a hierarchy of action abstraction and task decomposition. In this example, the high-level path plans are devised in **Level 1**, for efficiently fulfilling the tasks, e.g., bringing a child to the child center and going to the office by bike. Based on this plan, sub-tasks, such as following the bicycle lane and preventing collisions for safety reason, require decisions of action in **Level 2**, e.g., to go forward, turn left, or turn right. These tasks can be further decomposed down to a lower level, **Level 3**, such as sustaining the equilibrium of the bicycle and keeping a proper speed.

Main Research Question

How can aerospace systems exploit RL methods to improve the autonomy and online learning with respect to the a priori unknown system and environment, dynamical uncertainties, and partial observability?

This main research question is addressed in three specific methods and/or applications: (i) approximate dynamic programming with a quadratic cost function, (ii) adaptive critic designs, and (iii) high-level guidance and navigation.

ADP WITH QUADRATIC COST FUNCTION

LADP is an attractive combination of optimal control and RL strategy. Based on the assumption of LTI systems, this method uses temporary difference errors to numerically approximate the kernel matrix of the quadratic cost function. Therefore, this method is model-free, and the model information is implicitly contained in the kernel matrix. In addition, the method is efficient and mathematically explicit by calculating the optimal control input. However, most aerospace systems are nonlinear, and the eventually converged quadratic cost function cannot represent this nonlinearity. This challenge can be formulated as the first research question:

RQ1: How to generalize LADP to deal with nonlinear and/or time-varying systems, model mismatch, and partial observations, while retaining the efficiency and mathematical explicitness?

Adaptive Critic Designs

ACDs can be used in aerospace systems to control nonlinear systems by exploiting nonlinear function approximators. However, in ACDs, an accurate global system model still plays an important role. This model is identified off-line using representative simulation models, which may be difficult to obtain and are often not accurate themselves. In addition, the online adaptation of the system model also needs to be sufficiently quick and smooth, to deal with unforeseen dynamics in the system, uncertainties in the environment, and unexpected changes due to failures. This leads to the second research question:

RQ2: How to devise online ACDs and improve the online adaptability, to cope with internal uncertainties, external disturbances, and even sudden faults?

High-level Guidance and Navigation

With the increasing difficulty of high-level tasks, the state and action space can be high-dimensional, which even ADP methods cannot cope with. Recent research tackles the 'curse of dimensionality' in multi-objective, high-level decision-making problems through using hierarchical structures. However, the explicit rules of establishing the hierarchies and of assigning the rewards have not yet been well-published and usually involve engineer's preference, which may prevent the transfer learning from one application to another. Furthermore, current HRL methods often use the same or very similar RL methods within one application to ease the combination of different hierarchies. To acquire optimal decision-making efficiently, however, different levels within one HRL application often need different learning methods, learning types, rewards assignment, and even state information. The third research question of this dissertation, therefore, can be formulated as follows:

RQ3: How to establish a systematic HRL controller that deals with multiple objectives and partial observability, possesses transfer learning ability, and utilizes diverse RL methods?

1.3.2. Research Methods and Contributions

Incremental control methods [63–67] are often used in adaptive control to deal with system nonlinearity and uncertainty without identifying the global system. The incremental form of a nonlinear dynamic system is actually a linear time-varying approximation of the original system, assuming a sufficiently high sample rate for discretization. This form has been successfully applied to the design of nonlinear adaptive controllers, such as Incremental Nonlinear Dynamic Inversion (INDI) [63–66] and Incremental BackStepping (IBS) [67], to reduce their model dependency. Nevertheless, these methods still need some a priori knowledge of the system model, and have neither addressed optimization nor synthesis of designed closed-loop systems.

Therefore, the incremental control technique, in LADP and ACD methods, is used to generalize their use to nonlinear, unknown systems, and to improve their online adaptability. The main contributions in the optimal adaptive control field are listed as follows:

- A novel, model-free incremental Approximate Dynamic Programing (iADP) method is proposed for regulation problems with full state feedback and output feedback. This method uses incremental techniques to cope with system nonlinearities.

- An optimal tracking control method, based on iADP, is developed, to deal with unknown, time-varying internal dynamics and stochastic references with full state measurements and partial observability.

- This dissertation also presents the proofs and necessary conditions of the predictability of the system output in the regulation problem, and of the output tracking error in the tracking problem with partial observations.

- An *Incremental model* based Heuristic Dynamic Programming (IHDP) method is proposed to online and adaptively control unknown aerospace systems in the presence of nonlinear aerodynamic uncertainties, internal disturbances, and/or external disturbances.

- An *Incremental model* based Dual Heuristic Programming (IDHP) method is developed as an online ACD, which further improves the precision, accelerates the online learning, and deals with a wider range of initial conditions. This method is also validated to be successful in a Fault-Tolerant Control (FTC) task and in the presence of high-frequency measurement noise.

For high-level guidance and navigation, HRL methods have shown potential for large-scaled and complex tasks. The main contributions in this decision-making field are listed as follows:

- The *hybrid* Hierarchical Reinforcement Learning (hHRL) is proposed for online guidance and navigation in PO environment. This method allows for different learning methods, learning types, rewards assignment, and state information in different levels to improve the efficiency.

- The rules of establishing the hierarchies are set out to assimilate the multiple objectives and to allow transfer of learning within and across tasks.

1.3.3. Scope and Limitations

In order to focus on the main goal of this dissertation, the scope is limited as follows:

Aerospace systems: The proposed methods in this dissertation can be applied to, but are not necessarily limited to, aerospace systems. One application is an aerial vehicle, which is a second-order continuous missile model [68, 69]. This model is simple but nonlinear. It contains aerodynamic uncertainties and can operate at a high and rapidly changing angle of attack. It is suitable for a validation of the newly-developed model-free methods and for a fair comparison with the current RL methods. Another application is spacecraft attitude control disturbed with liquid sloshing [11, 12]. This is a Multiple-Input Multiple-Output (MIMO) nonlinear control problem, which is used to further validate the proposed methods and/or ideas in more complex systems and tasks. In the guidance and navigation part, the focus is shifted onto high-level decision-making problems. Therefore, the aerospace system is further simplified as a point mass model with discrete state and action spaces.

In addition, this dissertation focuses on the method development, theoretical analysis, and simulation experiments and does not include experiments on complex simulation models or any real systems.

Model-free: The proposed methods for low-level control all belong to model-free approaches, as they do not need any a priori information of the system dynamics nor online identification of the global system model. They assume a general continuous state space model, which can represent any aerospace system, and then identify the time-varying incremental model online to approximate the system linearized around the current instant.

Online reinforcement learning: Reinforcement learning is learning from the rewards/penalties, or even from failures. There may be several degrees of online learning ability requirements, which depend on the system stability, the reward/penalty assignment, and even the control objectives. If the system is inherently stable or has a representative model, the control policy can be updated iteratively until converged, and then used as an initial policy for mismatched systems or different control tasks with further online, recursive adaptation. On the other hand, if the system is open-loop unstable, and a priori unknown, online reinforcement learning needs to update the control policy online recursively and learns a feasible controller before failure. The online learning ability in one application can vary also, depending to the desired behaviors and reward assignments of each objective, such as preventing collisions and approaching the target in the guidance and navigation task. Because the system receives penalties after each collision, but rewards only after it reaches the target, the collision avoidance policy and goal reaching policy are updated within and after each iteration.

In addition, online learning also depends on the on-board computing capability, proper excitation and exploration, and other system features, which, however, are not addressed. This dissertation aims at enhancing the online applicability of current RL methods from the theoretical perspective.

Partial observability: Partial observability often occurs in aerospace systems when the system does not have enough information to infer all of its states [45]. Those methods dealing with deterministic systems and measurements are often referred to as output feedback methods [33, 46]. When stochastic, time-varying dynamics are involved, they belong to Partially Observable Markov Decision Processes (POMDPs) [36, 48, 49]. In low-level control tasks, stochastic, unknown dynamics, such as unpredictable gusts and unmeasurable, time-varying reference signals, bring stochastic dynamics into the measurement and may lead to partial observability. Note that in this situation the observability matrix of the system, from a control theoretic perspective, still can be full column rank. In guidance and navigation tasks, partial observability is often referred to as a non-perfect perception of the environment. The absolute state in the environment can not be inferred from the observation, e.g., in an indoor navigation task the flying robot is only equipped with limited visual sensors and it cannot perfectly know its absolute position.

1.4. Outline of the Thesis

The body of this thesis is divided into three parts to answer the three research questions, respectively, as seen in Fig. 1.4. Part I encompasses Chapters 2 and 3, which generalize the LADP methods to nonlinear systems with an approximate convex cost function. Part II, consisting of Chapters 4 and 5, introduces incremental models in ACDs to enhance the online applicability. Part III addresses a high-level guidance and navigation problem with a hybrid HRL method in Chapter 6. The outline of this thesis is as follows:

Chapter 2 proposes an effective and systematic adaptive control method for stabilization problems, called *incremental* Approximate Dynamic Programming (iADP) methods, to deal with system nonlinearity. This method combines the advantages of LADP methods and the incremental nonlinear control techniques to generate two model-free, effective adaptive flight controllers for nonlinear systems: iADP based on full-state feedback (iADP-FS) and iADP based on output feedback (iADP-OP). These two controllers are developed to solve optimal control problems with direct availability of full states and with only the availability of the system outputs.

In addition, Chapter 3 expands the idea of iADP to optimal tracking control problems for MIMO nonlinear systems and proposes two controllers for different observability conditions: full state measurement and partial observability. Because of the incremental model, the cost functions can be less dependent on the system or the reference and only need to be a rough approximation of the true cost-to-go. This approximation is a quadratic function only of the current tracking error, without expanding the dimension of the state space for the cost function to an augmented one. For each observability condition, two algorithms are developed for off-line batch learning and online recursive adaptation, respectively.

Chapter 4 develops an *Incremental model* based Heuristic Dynamic Programming (IHDP) method to deal with reference signal tracking problems. It generates a near-optimal controller for nonlinear systems, without a priori knowledge of the system dynamics. The IHDP method utilizes an online identified incremental model, instead of a neural network plant approximator, to simplify the updating of the actor network. This method can avoid off-line learning of the global system model, so as to improve the control performance and to accelerate the online learning efficiently.

Chapter 5 proposes another online self-learning adaptive controller for reference tracking problems, namely *Incremental model* based Dual Heuristic Programming (IDHP). This method accelerates the online learning compared to traditional DHP methods, and increases the convergence rate and control performance compared to the IHDP method presented in Chapter 4. In addition, this method is validated in an fault-tolerant control task and in the presence of measurement noise.

Chapter 6 designs a *hybrid* Hierarchical Reinforcement Learning (hHRL) method consisting of several levels, where each level uses different methods to optimize the learning with different state information and objectives. This method can help to accelerate learning, address the 'curse of dimensionality' in complex guidance and navigation tasks, reduce the uncertainty or ambiguity at higher levels, and efficiently transfer the learning results within and across tasks. The formulated rules of establishing the hierarchies make this method more flexible, transferable and closer to human behavior.

Chapter 7 concludes that (1) LADP methods can be applied to nonlinear systems using incremental techniques, while keeping them systematic and computationally efficient, (2) ACDs can utilize online identified incremental models to prevent off-line learning of the global model, to speed up the convergence rate, and to improve the control performance, and (3) hHRL methods provide a systematic design for guidance and navigation tasks with multiple objectives and partial observability.

1.5. Thesis Publications

This section lists the publication sources for main chapters:

- Chapter 2 is based on the following article:
  Y. Zhou, E. van Kampen, and Q. P. Chu, *Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback*, Journal of Guidance, Control, and Dynamics, Vol. 40, No. 2, p. 493-500, 2017. https://doi.org/10.2514/1.G001762.

- Chapter 3 is based on the following article:
  Y. Zhou, E. van Kampen, and Q. P. Chu, *Incremental approximate dynamic programming for nonlinear adaptive tracking control with partial observability*, Journal of Guidance, Control, and Dynamics, (under review).

- Chapter 4 is based on the following article:
  Y. Zhou, E. van Kampen, and Q. P. Chu, *Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming*, Acta Astronautica, (under review).

- Chapter 5 is based on the following article:
  Y. Zhou, E. van Kampen, and Q. P. Chu, *Incremental model based online dual heuristic programming for nonlinear adaptive control*, Control Engineering Practice, Vol. 73, p. 13-25, 2018. https://doi.org/10.1016/j.conengprac.2017.12.011.

- Chapter 6 is based on the following article:
  Y. Zhou, E. van Kampen, and Q. P. Chu, *Hybrid hierarchical reinforcement learning with partial observability*, Artificial Intelligence (submitted).

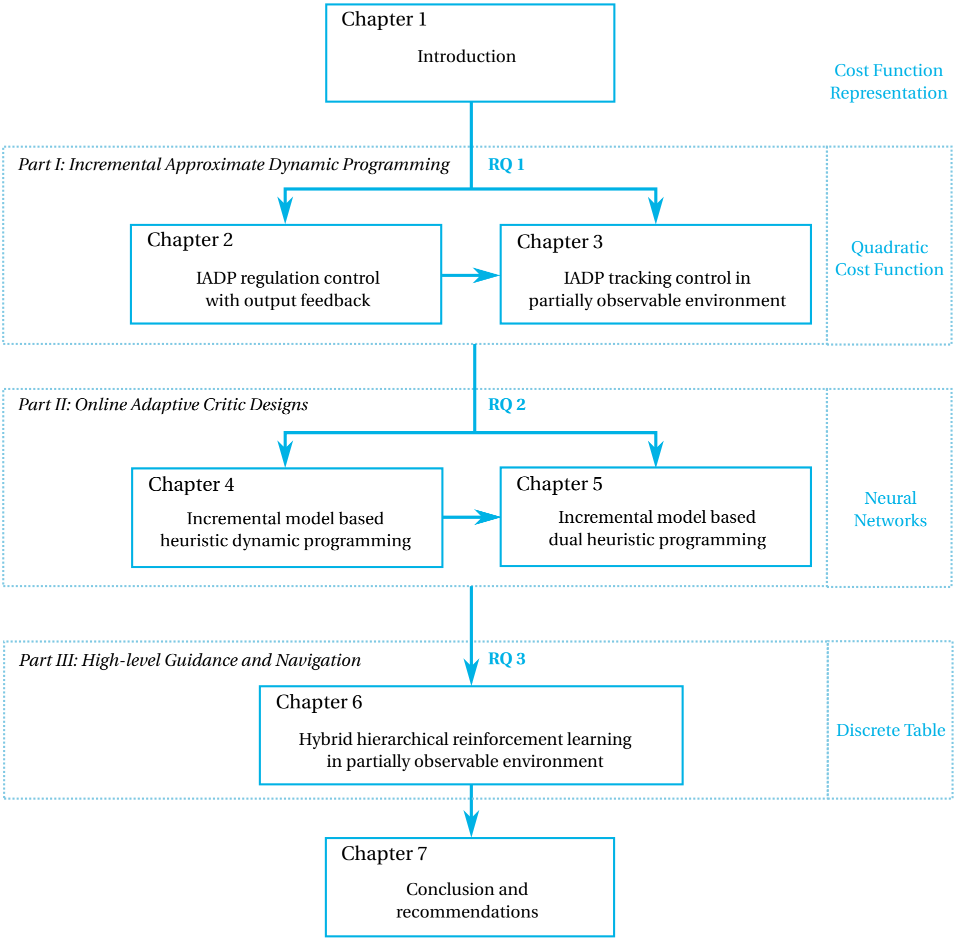


Figure 1.4: Outline of the thesis.

I

Incremental Approximate Dynamic Programming

2

Incremental Approximate Dynamic Programming for Regulation Control with Output Measurements

As was already suggested in the introduction, Linear Approximate Dynamic Programming (LADP) methods have attractive merits: model-free processes, mathematical explicitness, and efficiency of resource usage. However, these methods cannot be applied to nonlinear or time-varying systems. In this chapter, LADP methods are combined with incremental techniques to deal with nonlinear control problems. Two incremental Approximate Dynamic Programming (iADP) algorithms are developed, one which has direct availability of full states and one which uses only input/output measurements. This chapter starts with the development of these two iADP algorithms in Section 2.2. Section 2.3 validates these algorithms with numerical experiments on a simulated aerospace system. The results show improvement of the closed-loop performance of the nonlinear system.

This chapter is based on the following article:
Y. Zhou, E. van Kampen, and Q. P. Chu. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. Journal of Guidance, Control, and Dynamics. Vol. 40, No. 2, p. 493-500, 2017. https://doi.org/10.2514/1.G001762 [46].

2.1. Introduction

Model-free adaptive control approaches are worthwhile to be investigated for fault-tolerant flight control due to many unsolved challenges in model-based strategies [8, 9, 23–28]. Reinforcement Learning (RL) controllers have been proposed to solve non-linear, optimal control problems without using accurate system models [29, 31]. Traditional RL, solving optimality problems, is an off-line method using an n-dimensional look-up table for all possible state vectors, which may cause the "curse of dimensionality" [32, 34].

To tackle the "curse of dimensionality", numerical methods, such as Approximate Dynamic Programming (ADP), have been developed to solve the optimality problem [32, 44], by applying a function approximator with parameters to approximate the value/cost function. Searching for an applicable structure and parameters of the function approximator is a global optimization problem as these approximators are in general highly nonlinear. For special cases that the dynamics of the system are linear, Dynamic Programming (DP) gives a complete and explicit solution, because the one-step state cost and the cost function in these cases are quadratic [44]. For general nonlinear control problems, DP is difficult to carry out, and ADP designs are not systematic [34].

Considering the design challenges mentioned above, trade-off solutions which may lead to simple and systematic designs are extremely attractive. Some successful approaches have been reported lately [33, 70–72]. In this chapter, an incremental ADP (iADP) model-free adaptive control approach is developed for nonlinear systems. This control approach is inspired by the ideas and solutions given by several articles [33, 44, 63, 65, 67] . It starts with the selection of the cost function in a systematic way [44], and follows with the Linear ADP (LADP) model-free adaptive control approach [33]. As the plant to be controlled in this chapter is nonlinear, the iADP is developed based on the linearized incremental model of the original nonlinear system [63, 65, 67].

The incremental form of a nonlinear dynamic system is actually a linear time-varying approximation of the original system assuming sufficiently high sample rate for the discretization [63, 65, 67]. Combining LADP and the incremental form of the system to be controlled leads to a new nonlinear adaptive control algorithm iADP. It retains the advantages of LADP with a systematic formulation of cost function approximations for nonlinear systems, while keeping the closed-loop system optimized.

Classical ADP methods assume that the system is fully observable and that the observed states obey a Markov process. The problems of partial/imperfect information and unmeasurable state vector estimation are very challenging and demanded to be solved in numerous applications [45]. Many studies have already taken the presence of stochastic, time-varying wind disturbance into account as a general problem in practical navigation and guidance control [73, 74]. Despite that, parametrized output feedback controllers have been designed to deal with problems without full state information and to achieve finite time stability based on observers [38, 75–79]. However, these methods still need a priori knowledge and/or an assumption of the system model structure.

Other than that, output feedback ADP algorithms [33] have been proposed, as opposed to full state feedback, to tackle problems without direct state observations. These algorithms do not require any a priori knowledge of the system or engineering knowledge to design control parameters or even a separate observer. However, these algorithms are derived for affine in control input Linear Time-Invariant (LTI) systems.

The remainder of this chapter is structured as follows. Section 2.2 starts with an algorithm development combining ADP and the incremental approach assuming the direct availability of the full state observation [80], and follows by an iADP algorithm based on output feedback, which is designed by applying only the output and input measurement. Those algorithms are applied to a flight control simulation in section 2.3. Lastly, section 2.4 makes a brief conclusion on the benefits of using the proposed iADP methods as well as their limits and also addresses the challenges and possibilities for future work.

2.2. Incremental Approximate Dynamic Programming

Incremental methods are able to deal with nonlinear systems. These methods compute the required control increment at a certain moment using the conditions of the system in the instant before [65]. Aircraft models are highly nonlinear and can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t)], \tag{2.1}$$

$$\mathbf{y}(t) = h[\mathbf{x}(t)], \tag{2.2}$$

where $\mathbf{x} \in \mathscr{R}^n$ is the state vector, $\mathbf{u} \in \mathscr{R}^m$ is the control input, $\mathbf{y} \in \mathscr{R}^p$ is the measured output, $f[\mathbf{x}(t), \mathbf{u}(t)] \in \mathscr{R}^n$ provides the physical evaluation of n states over time, and $h[\mathbf{x}(t)] \in \mathscr{R}^p$ is the output (observation) function and can be measured using sensors.

The system dynamics around the condition of the system at time t_0 can be linearized by using the first-order Taylor series expansion:

$$\dot{\mathbf{x}}(t) \approx \dot{\mathbf{x}}(t_0) + F[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{x}(t) - \mathbf{x}(t_0)] + G[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{u}(t) - \mathbf{u}(t_0)], \tag{2.3}$$

where $F[\mathbf{x}(t), \mathbf{u}(t)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{x}(t)} \in \mathscr{R}^{n \times n}$ is the system matrix of the linearized model at time t, and $G[\mathbf{x}(t), \mathbf{u}(t)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{u}(t)} \in \mathscr{R}^{n \times m}$ is the control effectiveness matrix of the linearized model at time t.

It is assumed that the control inputs, states, and state derivatives of the system are measurable. Under this assumption, the model around time t_0 can be written in an incremental form:

$$\Delta\dot{\mathbf{x}}(t) \simeq F[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{x}(t) + G[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{u}(t). \tag{2.4}$$

This linearized incremental model is identifiable by using Least Square (LS) techniques.

2.2.1. Incremental Approximate Dynamic Programming Based on Full State Feedback

Physical systems are often continuous, but the collected data are discrete samples. It is assumed that the control system has a constant high sampling frequency. Thus, the nonlinear system can be written in a discrete form as follows:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{2.5}$$

$$\mathbf{y}_t = h(\mathbf{x}_t). \tag{2.6}$$

When the system has a direct availability of the full state observations, the output equation can be written as

$$\mathbf{y}_t = \mathbf{x}_t. \tag{2.7}$$

By taking the Taylor expansion, the linearized discrete model of this nonlinear system around $\mathbf{x}_{t-1}$, which approximates $\mathbf{x}_t$, can also be written in an incremental form:

$$\Delta\mathbf{x}_{t+1} \simeq F_{t-1}\Delta\mathbf{x}_t + G_{t-1}\Delta\mathbf{u}_t, \tag{2.8}$$

where $\Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$, $\Delta\mathbf{u}_t = \mathbf{u}_t - \mathbf{u}_{t-1}$, $F_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times n}$ is the system transition matrix, and $G_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times m}$ is the input distribution matrix at time step $t-1$. Because of the high frequency sample data and slow-varying system, the current linearized model (F_{t-1}, G_{t-1}) can be identified from L different data points using a piecewise sequential LS method [80, 81]. Because there are $n+m$ parameters in the ith row, L needs to satisfy $L \geq (n+m)$.

To minimize the cost of the system to reach its goal, the one-step cost function is defined quadratically:

$$c_t = c(\mathbf{y}_t, \mathbf{u}_t, \mathbf{y}_t^{ref}) = (\mathbf{y}_t - \mathbf{y}_t^{ref})^T Q(\mathbf{y}_t - \mathbf{y}_t^{ref}) + \mathbf{u}_t^T R\mathbf{u}_t, \tag{2.9}$$

where Q and R are positive definite matrices, and $\mathbf{y}_t^{ref}$ denotes the output reference. Considering a regulation control problem, the one-step cost function at time t can be written as

$$c_t = c(\mathbf{y}_t, \mathbf{u}_t) = \mathbf{y}_t^T Q\mathbf{y}_t + \mathbf{u}_t^T R\mathbf{u}_t. \tag{2.10}$$

For infinite horizons, the cost-to-go function is the cumulative future reward from any initial state $\mathbf{x}_t$:

$$\begin{aligned}\mathscr{J}^{\mu}(\mathbf{x}_t) &= \sum_{i=t}^{\infty} \gamma^{i-t}(\mathbf{y}_i^T Q\mathbf{y}_i + \mathbf{u}_i^T R\mathbf{u}_i)\\ &= \mathbf{y}_t^T Q\mathbf{y}_t + (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t)^T R(\mathbf{u}_{t-1} + \Delta\mathbf{u}_t) + \gamma\mathscr{J}^{\mu}(\mathbf{x}_{t+1}),\end{aligned} \tag{2.11}$$

where μ is the current *policy* (control law) for this iADP algorithm, $\gamma \in [0,1]$ is a parameter called the discounted rate or the forgetting factor. The cost-to-go function for the optimal policy μ^* is defined as follows:

$$\mathscr{J}^*(\mathbf{x}_t) = \min_{\Delta\mathbf{u}_t} \left[\mathbf{y}_t^T Q\mathbf{y}_t + (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t)^T R(\mathbf{u}_{t-1} + \Delta\mathbf{u}_t) + \gamma\mathscr{J}^*(\mathbf{x}_{t+1})\right]. \tag{2.12}$$

In this regulation problem, the policy μ is defined as the feedback control law in an incremental form:

$$\Delta\mathbf{u}_t = \mu(\mathbf{u}_{t-1}, \mathbf{x}_t, \Delta\mathbf{x}_t). \tag{2.13}$$

The optimal policy at time t is given by

$$\mu^* = \arg\min_{\Delta\mathbf{u}_t} \left[\mathbf{y}_t^T Q\mathbf{y}_t + (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t)^T R(\mathbf{u}_{t-1} + \Delta\mathbf{u}_t) + \gamma\mathscr{J}^*(\mathbf{x}_{t+1})\right]. \tag{2.14}$$

When the dynamics of the system are linear, this problem is known as the Linear-Quadratic Regulator (LQR) control problem. For this nonlinear case, the true cost-to-go is the sum of quadratic values in the outputs and inputs with a forgetting factor. Thus, the true cost-to-go $\mathscr{J}^{\mu}(\mathbf{x}_t)$ should always be positive. In general, ADP uses a surrogate cost function $\widehat{\mathscr{J}}^{\mu}(\mathbf{x}_t)$ to approximate the true cost-to-go. The goal is to capture its key features instead of accurately approximating the true cost-to-go. In many practical cases, even for time-varying systems, simple quadratic cost function approximations are chosen so that the evaluation step can be exactly carried out and the optimization problem is reduced to be tractable [44]. A systematic cost function approximation applied in this chapter is chosen to be quadratic in $\mathbf{x}_t$ for some symmetric, positive definite matrix $P \in \mathscr{R}^{n\times n}$:

$$\widehat{\mathscr{J}}^{\mu}(\mathbf{x}_t) = \mathbf{x}_t^T P \mathbf{x}_t. \tag{2.15}$$

This quadratic cost function approximation has an additional, important benefit for this approximately convex state-cost system with a fixed minimum value. To be specific, this system has an optimal state when it reaches the desired state and keeps it. The true cost-to-go may have local minima elsewhere because of the nonlinearity. On the other hand, this quadratic cost function has only one local minimum, which is also the global one. Therefore, this quadratic form helps to prevent the policy from going into any other local minimum. The learned symmetric, positive definite P matrix guarantees progressive optimization of the policy.

The LQR Bellman equation for $\widehat{\mathscr{J}}^{\mu}$ in the incremental form becomes

$$\begin{aligned}\widehat{\mathscr{J}}^{\mu}(\mathbf{x}_t) = {} & \mathbf{y}_t^T Q \mathbf{y}_t + (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t)^T R (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t) \\ & + \gamma(\mathbf{x}_t + F_{t-1}\Delta\mathbf{x}_t + G_{t-1}\Delta\mathbf{u}_t)^T P (\mathbf{x}_t + F_{t-1}\Delta\mathbf{x}_t + G_{t-1}\Delta\mathbf{u}_t).\end{aligned} \tag{2.16}$$

By setting the derivative with respect to $\Delta\mathbf{u}_t$ to zero, the *optimal control* can be obtained:

$$\Delta\mathbf{u}_t = -(R + \gamma G_{t-1}^T P G_{t-1})^{-1}[R\mathbf{u}_{t-1} + \gamma G_{t-1}^T P \mathbf{x}_t + \gamma G_{t-1}^T P F_{t-1}\Delta\mathbf{x}_t]. \tag{2.17}$$

From Eq. (2.17), it can be concluded that the policy is in the form of system variables $(\mathbf{u}_{t-1}, \mathbf{x}_t, \Delta\mathbf{x}_t)$ feedback, and the gains are functions of the dynamics of the current linearized system matrices (F_{t-1}, G_{t-1}).

Opposite to the model-based control algorithms with an online identification of global nonlinear systems, the availability of these local linear models is sufficient for iADP algorithms. Furthermore, the determination of the linear model structure is much simpler than the identification of the nonlinear model structure. If the nonlinear model is unknown, while the full state is measurable, the iADP algorithm, as shown below, can be applied to improve the policy iteratively online.

iADP algorithm based on Full State feedback (iADP-FS)

Evaluation. The cost function kernel matrix P under policy μ can be evaluated and updated recursively to Bellman equation for each iteration $j = 0, 1, \ldots$ until convergence:

$$\mathbf{x}_t^T P^{(j+1)} \mathbf{x}_t = \mathbf{y}_t^T Q \mathbf{y}_t + \mathbf{u}_t^T R \mathbf{u}_t + \gamma \mathbf{x}_{t+1}^T P^{(j)} \mathbf{x}_{t+1}. \tag{2.18}$$

Policy improvement. The policy improves for the new kernel matrix $P^{(j+1)}$:

$$\Delta\mathbf{u}_t = -(R + \gamma G_{t-1}^T P^{(j+1)} G_{t-1})^{-1}[R\mathbf{u}_{t-1} + \gamma G_{t-1}^T P^{(j+1)} \mathbf{x}_t + \gamma G_{t-1}^T P^{(j+1)} F_{t-1}\Delta\mathbf{x}_t]. \tag{2.19}$$

When Δt approximates to 0, the identified incremental model F_{t-1}, G_{t-1} and the prediction of the next state approximate their true values. With this linearized model, this problem locally becomes an LQR problem. Referring to optimal control problems, the policy designed above approaches the optimal policy as $\gamma = 1$. However, in ADP, the discount factor γ is usually chosen as $\gamma \in (0,1)$, so that the infinite sum has a finite value as long as the cost sequence is bounded, and the agent is not ‘myopic’ in being concerned only with maximizing immediate cost [31].

2.2.2. INCREMENTAL APPROXIMATE DYNAMIC PROGRAMMING BASED ON OUTPUT FEEDBACK

The full state of a system, such as an air vehicle system, is often not available. In addition, agents often try to control a system without enough information to infer its real states [45]. The Partially Observable Markov Decision Process (POMDP) framework can be used to deal with stochastic systems. For deterministic systems, these types of methods are often referred to as output feedback. The systems still need to be observable, which means that the unmeasurable internal states (the full state) can be reconstructed with the observations over a long enough time horizon. For model-free methods, the system is observable when the observability matrix has a full column rank.

Considering the nonlinear system again, see Eq. (2.5) and (2.6), the *output (observation)* around $\mathbf{x}_{t-1}$ can also be linearized with Taylor expansion:

$$\Delta \mathbf{y}_t \simeq H_{t-1}\Delta \mathbf{x}_t, \tag{2.20}$$

where $H_{t-1} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1}} \in \mathscr{R}^{p\times n}$ is the *observation matrix* at time step $t-1$. The nonlinear system incremental dynamics, see Eq. (2.8) and (2.20), at current time t can be represented by the previously measured data on time horizon [t-N, t]:

$$\Delta \mathbf{x}_t \simeq \widetilde{F}_{t-2,t-N-1}\cdot \Delta \mathbf{x}_{t-N} + U_N \cdot \overline{\Delta \mathbf{u}}_{t-1,t-N}, \tag{2.21}$$

$$\overline{\Delta \mathbf{y}}_{t,t-N+1} \simeq V_N \cdot \Delta \mathbf{x}_{t-N} + T_N \cdot \overline{\Delta \mathbf{u}}_{t-1,t-N}, \tag{2.22}$$

where symbol $\widetilde{F}_{t-a,t-b} = \prod_{i=t-a}^{t-b} F_i = F_{t-a}\cdot\,\cdots\,\cdot F_{t-b}$,

$$\overline{\Delta \mathbf{u}}_{t-1,t-N} = \begin{bmatrix} \Delta \mathbf{u}_{t-1} \\ \Delta \mathbf{u}_{t-2} \\ \vdots \\ \Delta \mathbf{u}_{t-N} \end{bmatrix} \in \mathscr{R}^{mN}, \quad \overline{\Delta \mathbf{y}}_{t,t-N+1} = \begin{bmatrix} \Delta \mathbf{y}_{t} \\ \Delta \mathbf{y}_{t-1} \\ \vdots \\ \Delta \mathbf{y}_{t-N+1} \end{bmatrix} \in \mathscr{R}^{pN},$$

$U_N = \begin{bmatrix} G_{t-2} & F_{t-2}G_{t-3} & \dots & \widetilde{F}_{t-2,t-N}\cdot G_{t-N-1} \end{bmatrix} \in \mathscr{R}^{n\times mN}$ is the *controllability matrix*,

$V_N = \begin{bmatrix} H_{t-1}\widetilde{F}_{t-2,t-N-1} \\ H_{t-2}\widetilde{F}_{t-3,t-N-1} \\ \vdots \\ H_{t-N}F_{t-N-1} \end{bmatrix} \in \mathscr{R}^{pN\times n}$ is the *observability matrix*,

$$T_N = \begin{bmatrix} H_{t-1}G_{t-2} & H_{t-1}F_{t-2}G_{t-3} & H_{t-1}\widetilde{F}_{t-2,t-3}G_{t-4} & \cdots & H_{t-2}\widetilde{F}_{t-3,t-N}\cdot G_{t-N-1} \\ 0 & H_{t-2}G_{t-3} & H_{t-2}F_{t-3}G_{t-4} & \cdots & H_{t-2}\widetilde{F}_{t-3,t-N}\cdot G_{t-N-1} \\ 0 & 0 & H_{t-3}G_{t-4} & \cdots & H_{t-3}\widetilde{F}_{t-4,t-N}\cdot G_{t-N-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & H_{t-N}\cdot G_{t-N-1} \end{bmatrix} \in \mathscr{R}^{pN\times mN}.$$

When the system is fully observable, the left inverse of V_N, which has a full column rank, can be obtained:

$$V_N^{left} = (V_N^T V_N)^{-1} V_N^T. \tag{2.23}$$

To have a full column rank for observability matrix V_N, N needs to satisfy $N \geq n/p$. Making the number of parameters to be identified as small as possible, the smallest value for N which meets $N \geq n/p$ is usually selected.

By left-multiplying V_N^{left} to Eq. (2.22), and then substituting the equation of $\Delta\mathbf{x}_{t-N}$ into Eq. (2.21), the incremental state can be reconstructed uniquely as a function of the past input/output:

$$\begin{aligned} \Delta\mathbf{x}_t &\simeq \widetilde{F}_{t-2,t-N-1}\cdot V_N^{left}\cdot\overline{\Delta\mathbf{y}}_{t,t-N+1} + (U_N - \widetilde{F}_{t-2,t-N-1}\cdot V_N^{left}\cdot T_N)\cdot\overline{\Delta\mathbf{u}}_{t-1,t-N} \\ &= \begin{bmatrix} M_{\Delta u} & M_{\Delta y}\end{bmatrix}\begin{bmatrix}\overline{\Delta\mathbf{u}}_{t-1,t-N} \\ \overline{\Delta\mathbf{y}}_{t,t-N+1}\end{bmatrix} \\ &= M_{t-1}\overline{\Delta\mathbf{z}}_{t,t-N}, \end{aligned} \tag{2.24}$$

where $M_{\Delta y}$ denotes $\widetilde{F}_{t-2,t-N-1}\cdot V_N^{left} \in \mathscr{R}^{n\times pN}$, $M_{\Delta u}$ denotes $(U_N - M_{\Delta y}T_N \in \mathscr{R}^{n\times mN})$, and $M_{t-1} = [M_{\Delta u}\ \ M_{\Delta y}] \in \mathscr{R}^{n\times(m+p)N}$. The matrix M_{t-1} is identifiable by using previous $\widehat{M}$ steps with $\widehat{M} \geq (m+p)N$.

The nonlinear incremental output equation, Eq. (2.20), can be represented by a history of measured input/output data on time horizon [t-N, t-1] in another form:

$$\overline{\Delta\mathbf{y}}_{t-1,t-N} \simeq \overline{V}_N\cdot\Delta\mathbf{x}_{t-N} + \overline{T}_N\cdot\overline{\Delta\mathbf{u}}_{t-1,t-N}, \tag{2.25}$$

where $\overline{V}_N = \begin{bmatrix} H_{t-2}\widetilde{F}_{t-3,t-N-1} \\ H_{t-3}\widetilde{F}_{t-4,t-N-1} \\ \vdots \\ H_{t-N-1}\end{bmatrix} \in \mathscr{R}^{pN\times n}$,

$$\overline{T}_N = \begin{bmatrix} 0 & H_{t-2}G_{t-3} & H_{t-2}F_{t-3}G_{t-4} & \cdots & H_{t-2}\widetilde{F}_{t-3,t-N}\cdot G_{t-N-1} \\ 0 & 0 & H_{t-3}G_{t-4} & \cdots & H_{t-3}\widetilde{F}_{t-4,t-N}\cdot G_{t-N-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & H_{t-N}\cdot G_{t-N-1} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathscr{R}^{pN\times mN}.$$

When the system is fully observable, the left inverse of $\overline{V}_N$, which also has a full column rank, can be obtained:

$$\overline{V}_N^{left} = (\overline{V}_N^T\overline{V}_N)^{-1}\overline{V}_N^T. \tag{2.26}$$

Left-multiplying $\overline{V}_N^{left}$ to Eq. (2.25) and substituting the resulted $\Delta\mathbf{x}_{t-N}$ into Eq. (2.21) and then the resulted $\Delta\mathbf{x}_t$ into Eq. (2.20), the dynamics of the output and of previous measured data can be obtained:

$$\begin{aligned}\Delta\mathbf{y}_t &\simeq (H_{t-1}U_N - H_{t-1}\widetilde{F}_{t-2,t-N-1}\cdot\overline{V}_N^{left}\overline{T}_N)\cdot\overline{\Delta\mathbf{u}}_{t-1,t-N}\\ &+ H_{t-1}\widetilde{F}_{t-2,t-N-1}\overline{V}_N^{left}\cdot\overline{\Delta\mathbf{y}}_{t-1,t-N}\\ &= \underline{G}_{t-1}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-N} + \underline{F}_{t-1}\cdot\overline{\Delta\mathbf{y}}_{t-1,t-N}.\end{aligned}\tag{2.27}$$

The output increment $\Delta\mathbf{y}_{t+1}$ can also be reconstructed uniquely as a function of the measured input/output data of N previous steps:

$$\begin{aligned}\Delta\mathbf{y}_{t+1} &\simeq \underline{G}_t\cdot\overline{\Delta\mathbf{u}}_{t,t-N+1} + \underline{F}_t\cdot\overline{\Delta\mathbf{y}}_{t,t-N+1}\\ &= \underline{G}_{t,11}\cdot\Delta\mathbf{u}_t + \underline{G}_{t,12}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-N+1} + \underline{F}_t\cdot\overline{\Delta\mathbf{y}}_{t,t-N+1},\end{aligned}\tag{2.28}$$

where $\underline{G}_t \in \mathscr{R}^{p\times Nm}$ is the extended control effectiveness matrix, $\underline{F}_t \in \mathscr{R}^{p\times Np}$ is the extended system matrix, $\underline{G}_{t,11} \in \mathscr{R}^{p\times m}$ and $\underline{G}_{t,12} \in \mathscr{R}^{p\times(N-1)m}$ are partitioned matrices from $\underline{G}_t$. Matrices $\underline{G}_t$ and $\underline{F}_t$ are identifiable by using the piecewise sequential LS method [80, 81]. In this case, there are $(m+p)N$ parameters in each row. Therefore, the number of previous data samples M needs to satisfy $M \geq (m+p)N$.

It is assumed that the cost function of the system state at time t can be written as a function of a symmetric expended kernel matrix $\overline{P} \in \mathscr{R}^{N(m+p)\times N(m+p)}$ in the quadratic form in terms of a history of observation vectors $\overline{\mathbf{z}}_{t,t-N} = [\overline{\mathbf{u}}_{t-1,t-N}^T, \overline{\mathbf{y}}_{t,t-N+1}^T]^T$:

$$\widehat{\mathscr{J}}^{\mu}(\mathbf{z}_{t,t-N}) = \overline{\mathbf{z}}_{t,t-N}^T\overline{P}\overline{\mathbf{z}}_{t,t-N}.\tag{2.29}$$

The optimal policy under the estimation of $\overline{P}$ in terms of $\overline{\mathbf{z}}_{t,t-N}$ is rewritten to be

$$\mu^* = \arg\min_{\Delta\mathbf{u}_t}(\mathbf{y}_t^TQ\mathbf{y}_t + \mathbf{u}_t^TR\mathbf{u}_t + \gamma\overline{\mathbf{z}}_{t+1,t-N+1}^T\,\overline{P}\,\overline{\mathbf{z}}_{t+1,t-N+1}),\tag{2.30}$$

where

$$\overline{\mathbf{z}}_{t+1,t-N+1}^T\,\overline{P}\,\overline{\mathbf{z}}_{t+1,t-N+1} = \begin{bmatrix}\mathbf{u}_{t-1}+\Delta\mathbf{u}_t\\ \overline{\mathbf{u}}_{t-1,t-N+1}\\ \mathbf{y}_t+\Delta\mathbf{y}_{t+1}\\ \overline{\mathbf{y}}_{t,t-N+2}\end{bmatrix}^T\begin{bmatrix}P_{11} & P_{12} & P_{13} & P_{14}\\ P_{12}^T & P_{22} & P_{23} & P_{24}\\ P_{13}^T & P_{23}^T & P_{33} & P_{34}\\ P_{14}^T & P_{24}^T & P_{34}^T & P_{44}\end{bmatrix}\begin{bmatrix}\mathbf{u}_{t-1}+\Delta\mathbf{u}_t\\ \overline{\mathbf{u}}_{t-1,t-N+1}\\ \mathbf{y}_t+\Delta\mathbf{y}_{t+1}\\ \overline{\mathbf{y}}_{t,t-N+2}\end{bmatrix}.\tag{2.31}$$

By differentiating with respect to $\Delta\mathbf{u}_t$, the policy improvement step can be obtained in terms of the measurements:

$$\begin{aligned}&-[R+\gamma P_{11}+\gamma(\underline{G}_{t,11})^T\cdot P_{33}\cdot\underline{G}_{t,11}+\gamma P_{13}\underline{G}_{t,11}+\gamma(P_{13}\underline{G}_{t,11})^T]\cdot\Delta\mathbf{u}_t\\ &=[R+\gamma P_{11}+\gamma(\underline{G}_{t,11})^T\cdot P_{13}^T]\mathbf{u}_{t-1}+\gamma[(\underline{G}_{t,11})^TP_{33}+P_{13}]\mathbf{y}_t\\ &\quad+\gamma[P_{12}+(\underline{G}_{t,11})^T\cdot P_{23}^T]\overline{\mathbf{u}}_{t-1,t-N+1}+\gamma[P_{14}+(\underline{G}_{t,11})^T\cdot P_{34}]\overline{\mathbf{y}}_{t,t-N+2}\\ &\quad+\gamma[(\underline{G}_{t,11})^TP_{33}+P_{13}](\underline{G}_{t,12}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-N+1}+\underline{F}_t\cdot\overline{\Delta\mathbf{y}}_{t,t-N+1}).\end{aligned}\tag{2.32}$$

If the nonlinear model is unknown, and only partial information about the states is accessible, the output feedback ADP algorithm combined with the incremental method can be applied to improve the policy iteratively online.

iADP algorithm based on OutPut feedback (iADP-OP)

Evaluation. The cost function kernel matrix $\overline{P}$ under policy μ can be evaluated and updated recursively according to Bellman equation for each iteration $j = 0, 1, \ldots$ until convergence:

$$\mathbf{z}'^{T}_{t,t-N+1}\overline{P}^{(j+1)}\mathbf{z}'_{t,t-N+1} = \mathbf{y}_t^T Q\mathbf{y}_t + \mathbf{u}_t^T R\mathbf{u}_t + \gamma\mathbf{z}'^{T}_{t+1,t-N+2}\overline{P}^{(j)}\mathbf{z}'_{t+1,t-N+2}. \tag{2.33}$$

Policy improvement. The policy improves for the new kernel matrix $\overline{P}^{(j+1)}$ according to the derived optimal control policy:

$$\begin{aligned}\Delta\mathbf{u}_t = &-[R+\gamma P_{11}+\gamma(\underline{G}_{t,11})^T\cdot P_{33}\cdot\underline{G}_{t,11}+\gamma P_{13}\underline{G}_{t,11}+\gamma(P_{13}\underline{G}_{t,11})^T]^{-1}\cdot\\ &\{[R+\gamma P_{11}+\gamma(\underline{G}_{t,11})^T\cdot P_{13}^T]\mathbf{u}_{t-1}+\gamma[(\underline{G}_{t,11})^T P_{33}+P_{13}]\mathbf{y}_t\\ &+\gamma[P_{12}+(\underline{G}_{t,11})^T\cdot P_{23}^T]\overline{\mathbf{u}}_{t-1,t-N+1}+\gamma[P_{14}+(\underline{G}_{t,11})^T\cdot P_{34}]\overline{\mathbf{y}}_{t,t-N+2}\\ &+\gamma[(\underline{G}_{t,11})^T P_{33}+P_{13}](\underline{G}_{t,12}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-N+1}+\underline{F}_t\cdot\overline{\Delta\mathbf{y}}_{t,t-N+1})\}.\end{aligned} \tag{2.34}$$

Approximating Δt to 0, the policy designed above approaches the optimal policy.

2.3. Numerical Experiments and Results

This section applies both iADP-FS and iADP-OF algorithms on a simulation of controlling an aerospace-related model, and shows how the algorithms perform in stabilizing and regulating the system in the presence of input disturbances and an initial offset.

2.3.1. Air vehicle model

A nonlinear air vehicle simulation model is used in this section. Air vehicle models can be highly nonlinear and are generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t) + \mathbf{w}(t)], \tag{2.35}$$

$$\mathbf{y}(t) = h[\mathbf{x}(t)], \tag{2.36}$$

where Eq. (2.35) is the kinematic state equation, $\mathbf{w}(t)$ is the external disturbance, which is set to be caused only by the input noise, Eq. (2.36) is the output (observation) equation.

As an application, only the elevator deflection will be regulated as pitch control to stabilize the air vehicle. Thus, two longitudinal states, angle of attack α and pitch rate q (i.e. $\mathbf{x} = [\alpha \quad q]$), and one control input, the elevator deflection angle δ_e, are concerned.

The nonlinear model in the pitch plane is simulated around a steady wings-level flight condition:

$$\dot{\alpha} = q + \frac{\bar{q}S}{m_a V_T} C_z(\alpha, q, M_a, \delta_e), \tag{2.37}$$

$$\dot{q} = \frac{\bar{q}Sd}{I_{yy}} C_m(\alpha, q, M_a, \delta_e), \tag{2.38}$$

where $\bar{q}$ is dynamic pressure, S is reference area, m_a is mass, V_T is speed, d is reference length, and I_{yy} is pitching moment of inertia. C_z and C_m are the aerodynamic force and moment coefficients, which are nonlinear functions.

The aerodynamic parameters of this model are valid for $-10° < \alpha < 10°$ [68, 69]:

$$\begin{aligned}
C_z(\alpha, q, M_a, \delta_e) &= C_{z1}(\alpha, M_a) + B_z\delta_e, \\
C_m(\alpha, q, M_a, \delta_e) &= C_{m1}(\alpha, M_a) + B_m\delta_e, \\
B_z &= b_1 M_a + b_2, \\
B_m &= b_3 M_a + b_4, \\
C_{z1}(\alpha, M_a) &= \phi_{z1}(\alpha) + \phi_{z2} M_a, \\
C_{z2}(\alpha, M_a) &= \phi_{m1}(\alpha) + \phi_{m2} M_a, \\
\phi_{z1}(\alpha) &= h_1\alpha^3 + h_2\alpha|\alpha| + h_3\alpha, \\
\phi_{m1}(\alpha) &= h_4\alpha^3 + h_5\alpha|\alpha| + h_6\alpha, \\
\phi_{z2} &= h_7\alpha|\alpha| + h_8\alpha, \\
\phi_{m2} &= h_9\alpha|\alpha| + h_{10}\alpha,
\end{aligned} \tag{2.39}$$

where $b_1, ..., b_4$, $h_1, ..., h_{10}$ are constant coefficients in the flight envelop, and the Mach number M_a is set to be 2.2.

When the input is $\mathbf{u}(t) = 0$, $\alpha = 0$ and $q = 0$ form an equilibrium of the system. The flight control task is to stabilize the system (i.e., a regulation problem), if there is any input disturbance or any offset from this condition. Specifically, an optimal policy μ^* and the associated optimum performance need to be found by minimizing the state cost [82].

2.3.2. Results

IADP algorithm based on full state feedback

As with other ADP methods, good state cost estimation depends heavily on the exploration of the state space, which is represented by persistent excitation in this case. An amplitude varying multiple doublet disturbance is used in this numerical experiment to test the performance of the proposed controllers. Figure 2.1 shows the response when an amplitude varying multiple doublet disturbance was used in the numerical example. The control system trained with the iADP algorithm rejects the disturbance compared to the response with the initial policy, where the initial kernel matrix is an identity matrix multiplied by a small positive value (0.01 in this chapter) $P_0 = 0.01I$.

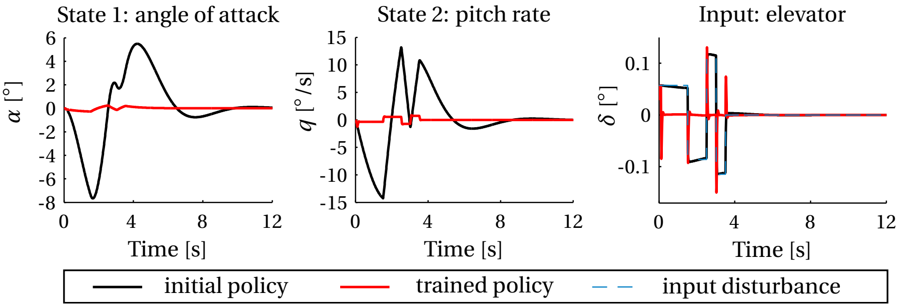


Figure 2.1: IADP-FS applied to nonlinear aircraft model with an amplitude varying multiple doublet disturbance.

Figure 2.2 shows the control performance when the initial state is an offset after a simulated gust. After training, the information of $G(\mathbf{x},\mathbf{u})$ and $F(\mathbf{x},\mathbf{u})$ can be used to estimate the current linearized system when the system cannot be identified using online identification without persistent excitation. Because the iADP method uses a simple quadratic cost function, the policy parameters of kernel matrix P converge after only two iterations.

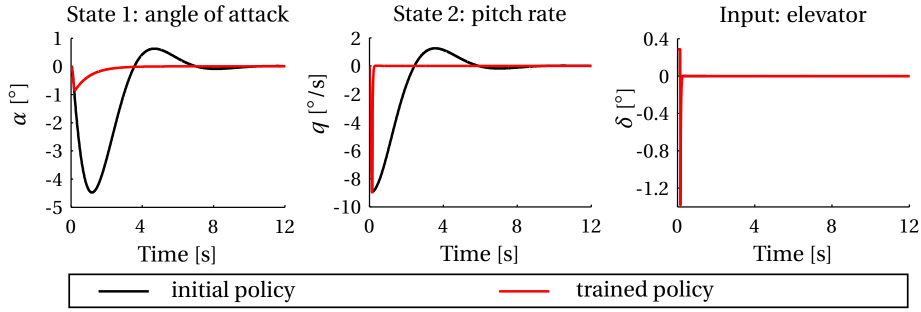


Figure 2.2: IADP-FS applied to nonlinear aircraft model with an initial offset.

This control method does not need the model of the nonlinear system, but still needs the full state to estimate the cost function and the control effectiveness matrix. If the model of the nonlinear system is unknown, and only coupled state information (observations) can be obtained, the iADP algorithm based on output feedback can be used.

IADP ALGORITHM BASED ON OUTPUT FEEDBACK

In practice, vane measurement techniques are cost-effective in measuring the angle of attack α [83]. Vanes are usually mounted on the aircraft in a location x_{vane} that allows for relatively undisturbed air flow to be measured:

$$\alpha_{measure} \simeq C_c(\alpha + \frac{x_{vane} - x_{cg}}{V} \cdot q), \tag{2.40}$$

where C_c denotes the calibration coefficient, and x_{cg} is the aircraft center of gravity. As a consequence, the kinematic position error induced by angular velocities q at the vane location has to be considered.

According to this practical case, the output/sensor measurement is set to be a combination of α and q with coefficients. Considering a practical case, which is to regulate α, a big portion of α (0.9) and a small portion of q (0.1) are selected:

$$\mathbf{y}(t) = [c_1\ c_2] \cdot \mathbf{x}(t) = [0.9\ 0.1] \cdot \begin{bmatrix} \alpha \\ q \end{bmatrix}. \tag{2.41}$$

Figure 2.3 shows the disturbance response when a 3211 input disturbance was introduced; Fig. 2.4 shows the control performance when the initial state is an offset from the stable condition after a simulated gust; Fig. 2.5 shows that the policy parameters of the kernel matrix converge quickly. After only 4 training iterations, the nonlinear system can be regulated, as shown in Figs. 2.3 and 2.4.

Note that when information of α is available, we can calculate q by using the identified model and enough previously measured observations. With some knowledge or assumptions on the model, the aircraft pitch plane system, α and q, is observable with only information of α. However, the proposed iADP algorithm is a model-free method, i.e., no assumptions about the model are needed, and the observations are from only two samples. Therefore, the observability is defined in terms of whether V_N in Eq. (2.22) has full column rank. If no information about one of the states can be provided, the iADP algorithm might not be beneficial.

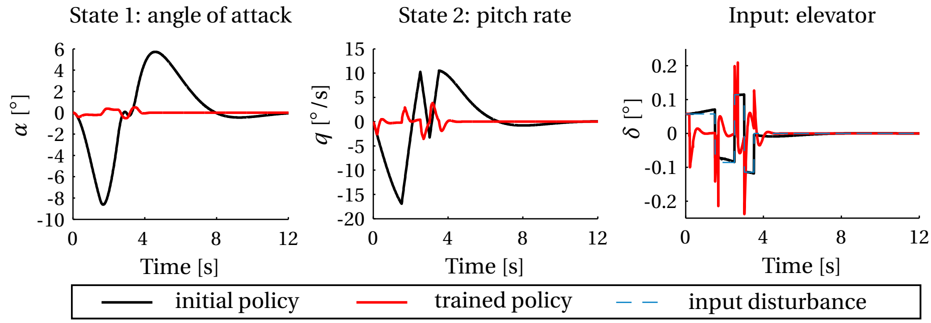


Figure 2.3: IADP-OP applied to nonlinear aircraft model with an amplitude varying multiple doublet disturbance.

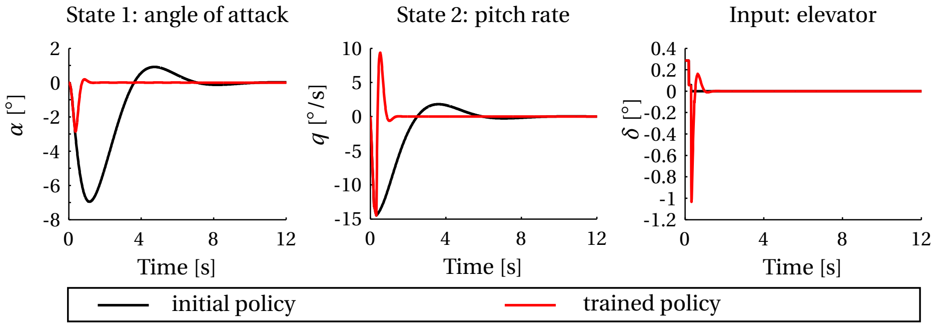


Figure 2.4: IADP-OP applied to nonlinear aircraft model with an initial offset.

Figures 2.6 and 2.7 show a comparison of disturbance response and natural response, respectively, among three policies. The initial policy is what the original system follows. It cannot compensate for the undesired inputs, such as gusts and ground effects. When the full state is available, the iADP-FS algorithm improves the closed-loop performance, lowers the disturbance response, and stabilizes the system from an offset much quicker. When the full state is unavailable, but the system is observable, the iADP-OP algorithm generates a policy. This policy has an almost equal ability to the policy of iADP-FS in terms of stabilizing and regulating the system.

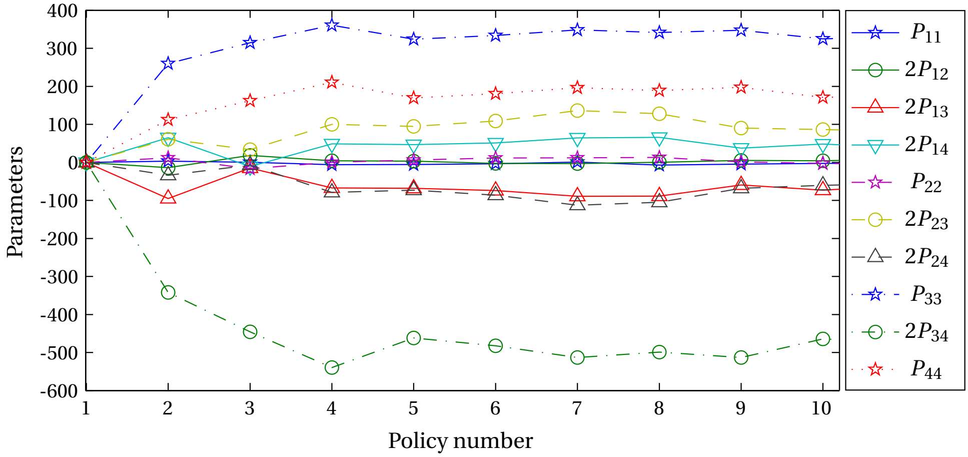


Figure 2.5: Kernel matrix parameters during training with IADP-OP.

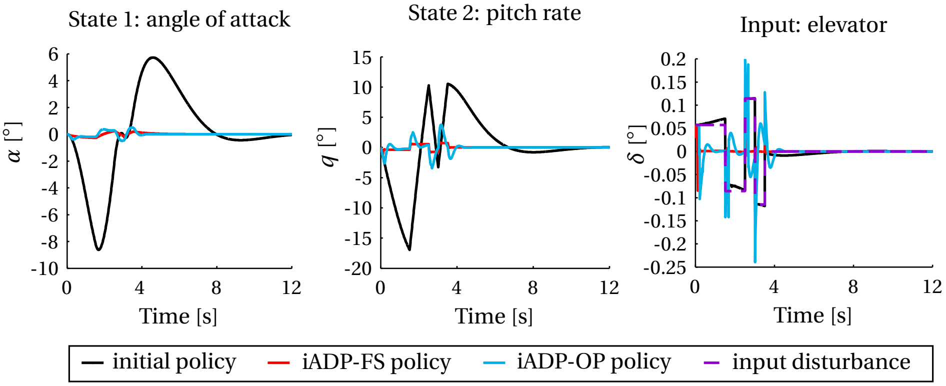


Figure 2.6: Comparison of policies applied to nonlinear aircraft model with an amplitude varying multiple doublet disturbance.

2.4. CONCLUSION

This chapter proposes a novel adaptive control method for nonlinear systems, called incremental Approximate Dynamic Programming (iADP). It systematically applies a quadratic cost function and greatly simplifies the design process of ADP. In addition, the incremental approach can deal with the nonlinearity of systems. The iADP method combines the advantages of both the Linear ADP (LADP) method and the incremental approach, and provides a model-free, effective adaptive flight controller for nonlinear systems. In addition to the iADP algorithm based on Full State feedback (iADP-FS), an iADP algorithm based on OutPut feedback (iADP-OP) is developed. IADP-OP uses only a history of input/output measurements from a dynamical nonlinear system to identify an extended incremental model.

Both the iADP-FS algorithm and the iADP-OP algorithm are applied to a missile model. The simulation results show that the trained policy with either algorithm rejects

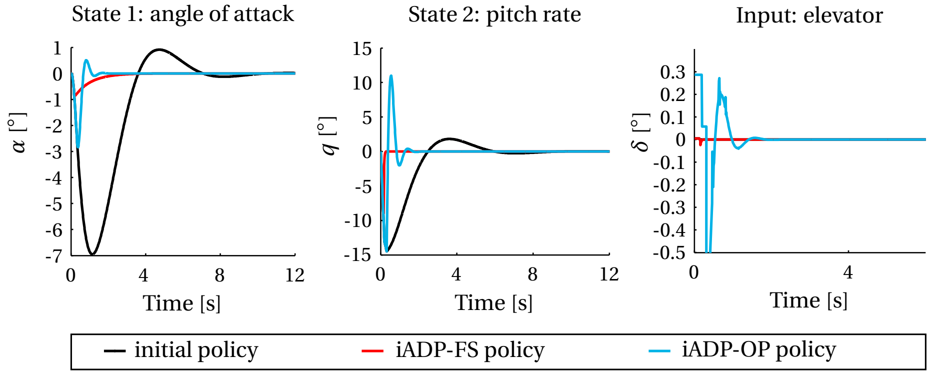


Figure 2.7: Comparison of policies applied to nonlinear aircraft model with an initial offset.

the disturbance compared to the response with the initial policy. This demonstrates that both the proposed algorithms improve the closed-loop performance of the nonlinear system, while keeping the design process simple and systematic as compared to conventional nonlinear ADP algorithms.

This new method can potentially design a near-optimal controller for nonlinear systems without a priori knowledge nor full state measurements of the dynamic model. Although still no theoretical guarantees on the nonlinear system performance can be offered, the performance of systems with approximately convex cost functions is observed to be very promising. For more complex systems, real applications, and general nonlinear control tasks, other possibilities such as piecewise quadratic cost functions will be studied in the future.

3

Incremental Approximate Dynamic Programming for Tracking Control with Partial Observability

The previous chapter presented the incremental Approximate Dynamic Programming (iADP) method for regulation problems. This chapter further expands the idea of iADP to tracking problems for multiple-input multiple-output nonlinear systems and to partially observable control problems. This chapter starts with the formulation of the iADP algorithms with full state observations and partial observability, in Section 3.2. Then, Section 3.3 describes the nonlinear spacecraft model, which has disturbance modeled by liquid sloshing. Section 3.4 validates the proposed method in tracking control tasks with both the full state observation and partial observability. For each observability condition, an off-line learning algorithm is applied to iteratively improve the policy until it is accurate enough, and hereafter an online algorithm is used to recursively update the policy in real-time. The results demonstrate that the proposed algorithms accurately and adaptively deal with time-varying internal dynamics while retaining efficient control, especially for unknown nonlinear systems with partial observability.

This chapter is based on the following article:
Y. Zhou, E. van Kampen, and Q. P. Chu. Adaptive spacecraft attitude control with incremental approximate dynamic programming. in 68th International Astronautical Congress (IAC) (Adelaide, Australia, 2017) [84].

Approximate dynamic programming is a class of reinforcement learning, which solves adaptive, optimal control problems and tackles the curse of dimensionality with function approximators. Within this category, linear approximate dynamic programming provides a model-free control method by systematically utilizing a quadratic cost function. Although efficient, linear approximate dynamic programming methods are difficult to apply to nonlinear or time-varying systems. To overcome the above limitations, this chapter proposes an adaptive nonlinear tracking control method based on incremental Approximate Dynamic Programming, which combines the advantages of linear approximate dynamic programming and incremental nonlinear control techniques. This is a model-free method for unknown, nonlinear systems and time-varying references. The trait of separating the local model information from the cost function approximation makes this method an option for partially observable control problems. This chapter, therefore, proposes two reference tracking controllers for different observability conditions: the direct full measurement, and the partial observation. In each condition, two algorithms are developed for off-line learning and online learning, respectively. These algorithms are applied to attitude control of a spacecraft disturbed by internal liquid sloshing. The results demonstrate that the proposed algorithms accurately deal with the unknown, time-varying internal dynamics while retaining efficient control, even with only partial observability.

3.1. Introduction

Adaptive control has been an active field of research for decades. It adapts the control law to the time-varying system or environment. Although model-based control strategies have been extensively studied and successfully applied to many applications [23–28, 85–88], they are heavily reliant on an accurate model or system identification. However, in many real applications, the system model may not be available, nor is its identification trivial [8, 9]. In addition, when uncertain dynamics or environments are involved, the mismatch between the model and real system may degrade the control performance of model-based methods. Therefore, this chapter aims to develop a model-free nonlinear adaptive control method in order to improve the tracking control performance.

Reinforcement Learning (RL) [29, 31] has been proposed to solve adaptive, optimal problems without using accurate system models. Conventional RL methods have successfully solved many problems in discrete spaces [31]. However, in the control field, applications often have continuous state and/or action spaces. To solve these problems, RL methods usually apply function approximators to tackle the ‘curse of dimensionality’. These methods are referred to as Approximate Dynamic Programming (ADP) methods. Adaptive Critic Designs (ACDs) constitute a class of ADP methods and have shown great success in optimal adaptive control of nonlinear problems [32, 34, 50, 51]. They are also well known as or Actor-Critic methods (ACs) because they separate the evaluation (critic) and improvement (actor) of the control policy using different parametric structures. The function approximation in ACDs are, mostly, implemented with black-box approaches such as Artificial Neural Networks (ANNs), which increase the complexity of analysis and design.

Opposite to ACDs is the Linear Approximate Dynamic Programming (LADP) method [33, 39, 89], which is another widely used model-free ADP method for linear systems with an explicit structure. This method utilizes a quadratic function to approximate the true cost-to-go and learns it adaptively. Based on this simple and efficient quadratic cost function, LADP can provide a mathematically explicit solution to linear optimal control problems [44]. Although model-free, efficient and online adaptive, LADP methods were devised based on the assumption that the dynamical system is Linear Time-Invariant (LTI). All the system information is implicitly contained in the quadratic cost function. These methods, therefore, have difficulties in solving problems with nonlinear or time-varying systems.

On the other hand, incremental control techniques can deal with system nonlinearity and uncertainties without identifying the global system [63–67]. The incremental model is a linear time-varying approximation of the original nonlinear dynamical system, assuming a sufficiently high sample rate for discretization. This technique has been successfully applied to adaptive nonlinear control methods, such as Incremental Nonlinear Dynamic Inversion (INDI) [63–66] and Incremental BackStepping (IBS) [67]. Although these methods have reduced the model dependency in the control system, optimization or synthesis of the designed closed-loop systems has not been addressed.

To combine the advantages of LADP methods and the incremental nonlinear control techniques, the incremental Approximate Dynamic Programming (iADP) method has been proposed to solve nonlinear regulation control problems off-line [46, 80, 81]. The regulation algorithms have been validated with a simple 2-state 1-input control problem for unknown, deterministic systems. This method, to some extend, can also be seen as a customized version of ACDs for convex cost function control problems [44, 90], or more specifically as Heuristic Dynamic Programming [53–55, 91, 92]. The quadratic cost function acts as the critic, the incremental model provides the model information, and thus, the actor is an explicit expression to optimal control. Although limited to convex cost function problems, this method does not need the identification of the global system or extensive tuning of nonlinear approximators.

However, real systems and control tasks are often more complex and involve unknown or uncertain dynamics, such as spacecraft attitude control with internal dynamics. Researchers have extensively studied non-adaptive nonlinear control methods for rigid satellites, or with flexible parts that can be modeled, such as solar panels. However, when unknown or uncertain dynamics are involved, the mismatch between the model and real system may degrade the performance of model-based methods.

Liquid sloshing is one of the unknown and uncertain internal dynamics interacting with the motion of the spacecraft [10–12]. A typical case is the liquid in the fuel tank sloshing around. The forces and torques acting on the spacecraft will slosh the fuel around. The fuel will, in turn, interact with the fuel tank and thus produce additional forces and torques to the spacecraft, which influence the performance of attitude control systems. Although fuel sloshing has been studied for years, an accurate liquid sloshing model is extremely difficult to obtain [12, 13].

This chapter, therefore, develops a systematic optimal adaptive tracking control method based on the idea of iADP and generates off-line and online model-free control algorithms for unknown, time-varying dynamical systems. This method belongs to model-free control because it does not need any a priori information of the system dynamics, online identification of the global nonlinear system, nor even an assumption of the time scale separation, but only an incremental model. The increment model, in this chapter, is identified online using a Recursive Least Square (RLS) technique assuming a sufficiently high sample rate. The property of separating this local model information from the cost function approximation makes iADP suitable for Partially Observable (PO) control problems.

Partial observability often happens in real applications, when the system does not have enough information to infer its real states [45]. Those methods dealing with deterministic systems and measurements are often referred to as output feedback methods [33, 46, 47, 76, 93]. When stochastic, time-varying dynamics are involved, it belongs to Partially Observable Markov Decision Process (POMDP) [36, 48, 49]. This chapter also develops off-line and online algorithms for optimal tracking problems in PO conditions. The only measurement is the output tracking error, where the unknown, time-varying reference signal brings the stochastic dynamics into the measurement.

This proposed PO algorithms can be applied to spacecraft rendezvous/docking problems [93–95] with liquid sloshing in the chaser, such as the Automated Transfer Vehicle (ATV) [96, 97], a cargo ship to the International Space Station (ISS). Liquid sloshing may happen due to the 4,700 kg propellant for guidance and control of the vehicle with a launch mass of 20,750 kg. The current docking sensors are infrared cameras, which only provide the navigation information about the tracking errors between the ATV and the ISS. Due to the flexibility of ISS, ATV has to dock to the moving or vibrating docking port of the Russian service module on the ISS. These challenges in aerospace systems and tasks motivate the development of the new control approach proposed in this chapter.

The remainder of this chapter is structured as follows. Section 3.2 presents the formulation of the iADP algorithms with full state observations and partial observability. Section 3.3 introduces the Multiple-Input Multiple-Output (MIMO) nonlinear spacecraft model disturbed with liquid sloshing and discusses the implementation of the proposed control method. Then, section 3.4 validates the algorithms with both the full state observation and partial observability by applying the off-line algorithms to the tracking problem on one model, and then on another mismatched model and/or different reference signal with online adaptation. Lastly, section 3.5 shows the advantages and disadvantages of using the iADP and addresses the challenges and possibilities for future research.

3.2. Incremental Approximate Dynamic Programming for Tracking Control

The incremental Approximate Dynamic Programming (iADP) method expands the LADP to solve nonlinear control problems with an incremental model. It was first proposed to solve regulation problems for nonlinear systems [46, 80, 81]. In this chapter, this idea will be further developed for more complex systems and more general control tasks with different observation conditions. This section first introduces the idea of the incremental approach, and then presents and elaborates the proposed iADP method for tracking problems both with full state measurements and with only partial observations.

3.2.1. The Incremental Approach

The incremental approach is approximating the nonlinear system with a time-varying linear model. This incremental model is identified at each moment by using the conditions of the system in an instant before [65]. This technique has been successfully applied to deal with unknown or inaccurate nonlinear systems [46, 63–67, 80, 81, 91]. The dynamic and kinematic equations of a nonlinear system can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t)], \tag{3.1}$$

where $\mathbf{x} \in \mathscr{R}^n$ is the system state vector, $\mathbf{u} \in \mathscr{R}^m$ is the control input vector, and $f[\mathbf{x}(t), \mathbf{u}(t)] \in \mathscr{R}^n$ provides the physical evaluation of the state vector over time.

If the system is first-order continuous, the system dynamics around the condition at time t_0 can be linearized approximately by using the first-order Taylor series expansion:

$$\dot{\mathbf{x}}(t) \approx \dot{\mathbf{x}}(t_0) + F[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{x}(t) - \mathbf{x}(t_0)] + G[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{u}(t) - \mathbf{u}(t_0)], \tag{3.2}$$

where $F[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{x}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)} \in \mathscr{R}^{n\times n}$ is the system matrix and $G[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{u}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)} \in \mathscr{R}^{n\times m}$ is the control effectiveness matrix of the linearized model at time t_0.

When the control inputs, states, and state derivatives of the system are measurable, i.e., $\Delta\mathbf{u}$, $\Delta\dot{\mathbf{x}}$, and $\Delta\mathbf{x}$ are measurable, the system model around time t_0 can be written in an incremental form:

$$\Delta\dot{\mathbf{x}}(t) \simeq F[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{x}(t) + G[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{u}(t). \tag{3.3}$$

Although most physical systems are continuous, their measurements are often discrete. With a sufficiently high, constant data sampling frequency, the before-mentioned nonlinear systems can also be written in a discrete form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t). \tag{3.4}$$

The system dynamics around $\mathbf{x}_t$ can also be linearized by taking the Taylor expansion, as follows:

$$\mathbf{x}_{t+1} \approx \mathbf{x}_t + \mathbf{F}_{t-1}\cdot(\mathbf{x}_t - \mathbf{x}_{t-1}) + \mathbf{G}_{t-1}\cdot(\mathbf{u}_t - \mathbf{u}_{t-1}), \tag{3.5}$$

where $\mathbf{F}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times n}$ is the system transition matrix, and $\mathbf{G}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times m}$ is the input distribution matrix at time step $t-1$ for discretized systems. The incremental form of this discrete nonlinear system can be written as

$$\Delta\mathbf{x}_{t+1} \approx \mathbf{F}_{t-1}\Delta\mathbf{x}_t + \mathbf{G}_{t-1}\Delta\mathbf{u}_t. \tag{3.6}$$

The nonlinear system can be represented as the time-varying incremental model. This linear model needs to be available online to provide the model information for the iADP algorithm instead of using a global nonlinear system model. With the high-frequency sample data and the relatively slow-varying system assumption, time-varying matrices $\mathbf{F}_{t-1}$ and $\mathbf{G}_{t-1}$ can be identified online using Least Squares (LS) techniques.

3.2.2. IADP with Full State Feedback

Since the iADP method assumes a time-varying incremental model, it can be used to solve MIMO tracking problems. In addition, this method optimizes the control increment by minimizing the system cost approaching its goal. Therefore, it does not assume the time-scale separation, but need to define the one-step cost function:

$$c(t) = (\mathbf{x}_t - \mathbf{x}_t^{ref})^T Q(\mathbf{x}_t - \mathbf{x}_t^{ref}) + \mathbf{u}_t^T R\mathbf{u}_t, \tag{3.7}$$

where Q and R are positive definite matrices, and $\mathbf{x}_t^{ref}$ is the reference signal for the system state. The cost-to-go function from any initial state $\mathbf{x}_t$ and reference $\mathbf{x}_t^{ref}$ under current policy μ is the cumulative sum of future costs:

$$\begin{aligned} J^{\mu}(t) &= \sum_{l=t}^{\infty} \gamma^{l-t}[(\mathbf{x}_l - \mathbf{x}_l^{ref})^T Q(\mathbf{x}_l - \mathbf{x}_l^{ref}) + \mathbf{u}_l^T R\mathbf{u}_l] \\ &= (\mathbf{x}_t - \mathbf{x}_t^{ref})^T Q(\mathbf{x}_t - \mathbf{x}_t^{ref}) + \mathbf{u}_t^T R\mathbf{u}_t + \gamma J^{\mu}(t+1), \end{aligned} \tag{3.8}$$

where $\gamma \in (0,1)$ is the forgetting factor representing the importance of the cost in the future. This nonlinear tracking control defines its one-step cost quadratically in the tracking error $\mathbf{e}_t = \mathbf{x}_t - \mathbf{x}_t^{ref}$ and the control input $\mathbf{u}_t$. Therefore, the true cost-to-go, which is the sum of these quadratic values with a forgetting factor, should always be positive.

To represent the true cost-to-go, the LADP method uses a cost function of the augmented state [89, 98] consisting of the system states and the references for Linear Quadratic Tracking (LQT) problems. The reason is that the conventional LADP method does not have a separate structure to approximate the model, and all the linear model information are included in the cost function. The iADP method, on the other hand, assumes the system to be time-varying and uses an online identified incremental model to deal with the changes of the system. Therefore, the true cost-to-go can be approximated by a function of the tracking error $\mathbf{e}_t$.

ADP methods, in general, use a surrogate function to approximate the true cost-to-go so as to capture the key attributes or features instead of the accurate values. As with many other practical cases, LADP uses simple quadratic cost functions, with which the expectation step can be explicitly carried out, and the optimization problem becomes tractable [44]. The iADP method, therefore, adopts this quadratic cost function in the tracking error $\mathbf{e}_t$ for a symmetric, positive definite kernel matrix $P \in \mathscr{R}^{n\times n}$, as shown below:

$$\widehat{J}^{\mu}(\mathbf{e}_t) = \mathbf{e}_t^T P\mathbf{e}_t. \tag{3.9}$$

Another benefit of the above quadratic cost function is that it only has one local/global minimum, although the true cost-to-go may have many local minima due to the nonlinearity of the system and the uncertainty of the tracking task. Specifically, this system has an optimal state at each moment when satisfying the condition $\mathbf{e}_t = 0$. Therefore, this quadratic form helps to prevent the policy from going into any other local minimum. To guarantee the progressive optimization of the policy, the P matrix learned from proper data samples should be symmetric, positive definite.

Except for the cost function, the optimization of the iADP method also depends on the tracking error prediction. With the incremental model information $\mathbf{F}_{t-1}$ and $\mathbf{G}_{t-1}$, the tracking error can be predicted approximately as follows:

$$\begin{aligned}\mathbf{e}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}_{t+1}^{ref} \\ &\approx \mathbf{x}_t + \mathbf{F}_{t-1}\Delta\mathbf{x}_t + \mathbf{G}_{t-1}\Delta\mathbf{u}_t - \mathbf{x}_t^{ref} - \Delta\mathbf{x}_{t+1}^{ref} \\ &= \mathbf{e}_t + \mathbf{F}_{t-1}\Delta\mathbf{x}_t + \mathbf{G}_{t-1}\Delta\mathbf{u}_t - \Delta\mathbf{x}_{t+1}^{ref}.\end{aligned} \tag{3.10}$$

The reference dynamics are often taken into consideration for tracking control problems. In the full state observable condition, all the states and references are measurable. In this situation, an assumption is made as follows:

Assumption 1. *The reference signal is slow-variant, and the sampling frequency is sufficiently high.*

Under this assumption, the increment of the reference $\Delta\mathbf{x}_{t+1}^{ref}$ can be ignored. In another word, the control input $\mathbf{u}_t$ is designed to track the current, available reference signal $\mathbf{x}_t^{ref}$. Therefore, the tracking error at the next time instant can be further approximated as follows:

$$\mathbf{e}_{t+1} \approx \mathbf{e}_t + \mathbf{F}_{t-1}\Delta\mathbf{x}_t + \mathbf{G}_{t-1}\Delta\mathbf{u}_t. \tag{3.11}$$

Consequently, the Bellman equation for $\widehat{J}^{\mu}$ becomes

$$\begin{aligned}\widehat{J}^{\mu}(\mathbf{e}_t) &= \mathbf{e}_t^T Q\mathbf{e}_t + \mathbf{u}_t^T R\mathbf{u}_t + \gamma\mathbf{e}_{t+1}^T P\mathbf{e}_{t+1} \\ &\approx \mathbf{e}_t^T Q\mathbf{e}_t + \mathbf{u}_t^T R\mathbf{u}_t + \gamma(\mathbf{e}_t + \mathbf{F}_{t-1}\Delta\mathbf{x}_t + \mathbf{G}_{t-1}\Delta\mathbf{u}_t)^T P(\mathbf{e}_t + \mathbf{F}_{t-1}\Delta\mathbf{x}_t + \mathbf{G}_{t-1}\Delta\mathbf{u}_t).\end{aligned} \tag{3.12}$$

In addition, the optimal cost-to-go $J^*(t)$ is defined as

$$J^*(t) = \min_{\Delta\mathbf{u}_t}[\mathbf{e}_t^T Q\mathbf{e}_t + (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t)^T R(\mathbf{u}_{t-1} + \Delta\mathbf{u}_t) + \gamma J^*(t+1)]. \tag{3.13}$$

Thus, the optimal policy μ^* at time t can be given by

$$\mu^* = \arg\min_{\Delta\mathbf{u}_t}[\mathbf{e}_t^T Q\mathbf{e}_t + (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t)^T R(\mathbf{u}_{t-1} + \Delta\mathbf{u}_t) + \gamma J^*(t+1)]. \tag{3.14}$$

With this optimality principle, the optimal control input can be derived by setting the derivative of the approximated cost function $\widehat{J}^{\mu}(\mathbf{e}_t)$ in Eq. (3.12) w.r.t. $\Delta\mathbf{u}_t$ to zero, as follows:

$$\begin{aligned}\Delta\mathbf{u}_t &= -(R + \gamma\mathbf{G}_{t-1}^T P\mathbf{G}_{t-1})^{-1}[R\mathbf{u}_{t-1} + \gamma\mathbf{G}_{t-1}^T P(\mathbf{e}_t + \mathbf{F}_{t-1}\Delta\mathbf{x}_t)] \\ &= -(R + \gamma\mathbf{G}_{t-1}^T P\mathbf{G}_{t-1})^{-1}[R\mathbf{u}_{t-1} + \gamma\mathbf{G}_{t-1}^T P\mathbf{e}_t + \gamma\mathbf{G}_{t-1}^T P\mathbf{F}_{t-1}\Delta\mathbf{x}_t].\end{aligned} \tag{3.15}$$

From Eq. (3.15), we can conclude that the policy is in the form of system variables $(\mathbf{u}_{t-1}, \mathbf{e}_t, \Delta\mathbf{x}_t)$ feedback, and the gains are functions of the dynamics of the current linearized system $(\mathbf{F}_{t-1}, \mathbf{G}_{t-1})$ and kernel matrix P.

If the nonlinear model is unknown, while the Full State (FS) is measurable, the iADP method can be applied to improve the tracking control policy as follows:

Batch iADP-FS algorithm for reference tracking

Evaluation. The cost function kernel matrix P under policy μ can be evaluated and updated recursively with the Bellman equation for each iteration $j = 0, 1, \ldots$ until convergence:

$$\mathbf{e}_t^T P^{(j+1)} \mathbf{e}_t = \mathbf{e}_t^T Q \mathbf{e}_t + \mathbf{u}_t^T R \mathbf{u}_t + \gamma \mathbf{e}_{t+1}^T P^{(j)} \mathbf{e}_{t+1}. \tag{3.16}$$

Policy improvement. The policy improves for the new kernel matrix $P^{(j+1)}$:

$$\Delta \mathbf{u}_t = -(R + \gamma \mathbf{G}_{t-1}^T P^{(j+1)} \mathbf{G}_{t-1})^{-1} [R \mathbf{u}_{t-1} + \gamma \mathbf{G}_{t-1}^T P^{(j+1)} \mathbf{e}_t + \gamma \mathbf{G}_{t-1}^T P^{(j+1)} \mathbf{F}_{t-1} \Delta \mathbf{x}_t]. \tag{3.17}$$

Opposite to model-based control algorithms with an online identification of global nonlinear systems, the current approach needs only a local linear model. In another word, availability of the local linear model is sufficient for iADP algorithms. This feature considerably reduces the complexity of the nonlinear control problems because the determination of the linear model structure is much simpler than the identification of the nonlinear model structure.

Although the progressive optimization of the policy should theoretically be guaranteed, the convergence of the batch iADP algorithms, as with other ADP methods, depends on the data samples and the state space exploration. When the trained policy $P^{(j)}$ can track the reference very accurately, and the input disturbances are rejected, the system does not have much exploration, and the parameters in $P^{(j+1)}$ might not be identified using the batch LS. Therefore, when the averaged one-step-cost in an iteration j is smaller than a threshold $\bar{c}(t)^{(j)} < c_{threshold}$, the kernel matrix P will be updated using RLS with forgetting factor $\gamma_P^{RLS} = 1$. The iADP-FS algorithm in Eqs. (3.16) and (3.17) will be rewritten using the RLS method as follows:

Recursive iADP-FS algorithm for reference tracking

Evaluation. The cost function kernel matrix P within iteration $j+1$ can be evaluated and updated recursively for each time step t:

$$\mathbf{e}_t^T P_t \mathbf{e}_t = \mathbf{e}_t^T Q \mathbf{e}_t + \mathbf{u}_t^T R \mathbf{u}_t + \gamma \mathbf{e}_{t+1}^T P_{t-1} \mathbf{e}_{t+1}. \tag{3.18}$$

Policy improvement. The policy improves for the new kernel matrix P_t:

$$\Delta \mathbf{u}_t = -(R + \gamma \mathbf{G}_{t-1}^T P_t \mathbf{G}_{t-1})^{-1} [R \mathbf{u}_{t-1} + \gamma \mathbf{G}_{t-1}^T P_t \mathbf{e}_t + \gamma \mathbf{G}_{t-1}^T P_t \mathbf{F}_{t-1} \Delta \mathbf{x}_t]. \tag{3.19}$$

In iteration $j+1$, the initial kernel matrix P_0 is the last updated $P^{(j)}$ from the previous iteration. This recursive algorithm will also be used when the off-line trained policy are applied to the real system online.

3.2.3. IADP WITH PARTIAL OBSERVABILITY

In many real applications, such as object tracking with visual sensors, the only measurement is the output tracking error, which is partially observed relative state:

$$\mathbf{e}_{t+1} = \mathbf{y}_{t+1} - \mathbf{y}_{t+1}^{ref}, \tag{3.20}$$

where $\mathbf{y}_{t+1} \in \mathscr{R}^p$ is the output of the nonlinear system, and $\mathbf{y}_{t+1}^{ref} \in \mathscr{R}^p$ is the output reference. This measurement correlated with the reference signal introduces the stochastic dynamics into this tracking problem. Therefore, this problem can be seen as a Partially Observable (PO) control problem.

The output of the nonlinear system can be written as

$$\mathbf{y}_{t+1} = h(\mathbf{x}_{t+1}), \tag{3.21}$$

where h denotes the output equation. The output around time instance t can also be linearized with Taylor expansion:

$$\Delta\mathbf{y}_{t+1} = H_t \Delta\mathbf{x}_{t+1}, \tag{3.22}$$

where $H_t = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}_{t+1}} \in \mathscr{R}^{p\times n}$ is the observation matrix at the current time instance.

The first step of applying iADP methods is to analyze the predictability of the tracking error. It has been approved that if a nonlinear system is fully observable with its output, this system, especially for regulation problems, still can be seen as deterministic [46]. The system output can, therefore, be represented by the previous input/output measurements over a long-enough time horizon $[t-N, t]$ as derived in [46] :

$$\begin{aligned}\Delta\mathbf{y}_{t+1} &\approx \underline{G}_t \cdot \overline{\Delta\mathbf{u}}_{t,t-N+1} + \underline{F}_t \cdot \overline{\Delta\mathbf{y}}_{t,t-N+1} \\ &= \underline{G}_{t,11} \cdot \Delta\mathbf{u}_t + \underline{G}_{t,12} \cdot \overline{\Delta\mathbf{u}}_{t-1,t-N+1} + \underline{F}_t \cdot \overline{\Delta\mathbf{y}}_{t,t-N+1},\end{aligned} \tag{3.23}$$

where $N \geq n/p$, $\underline{G}_t \in \mathscr{R}^{p\times Nm}$, $\underline{F}_t \in \mathscr{R}^{p\times Np}$, $\underline{G}_{t,11} \in \mathscr{R}^{p\times m}$ and $\underline{G}_{t,12} \in \mathscr{R}^{p\times(N-1)m}$ are partitioned matrices from $\underline{G}_t$. Matrices $\underline{G}_t$ and $\underline{F}_t$ are identifiable by using LS methods with output/input measurements [46, 80].

If accepting the assumption from the previous section (section 3.2.2) that the output reference signal $\mathbf{y}^{ref}$ is slow-variant and the sampling frequency is sufficiently high, the increment of the reference can be ignored. Therefore, the tracking error at the next time instant can be approximated as follows:

$$\begin{aligned}\mathbf{e}_{t+1} &= \mathbf{y}_{t+1} - \mathbf{y}^{ref}_{t+1} \\ &\approx \mathbf{y}_t + \underline{F}_t \overline{\Delta\mathbf{y}}_{t,t-N+1} + \underline{G}_t \overline{\Delta\mathbf{u}}_{t,t-N+1} - \mathbf{y}^{ref}_t - \Delta\mathbf{y}^{ref}_{t+1} \\ &\approx \mathbf{e}_t + \underline{F}_t \overline{\Delta\mathbf{y}}_{t,t-N+1} + \underline{G}_t \overline{\Delta\mathbf{u}}_{t,t-N+1} \\ &\approx \mathbf{e}_t + \underline{F}_t \overline{\Delta\mathbf{e}}_{t,t-N+1} + \underline{G}_t \overline{\Delta\mathbf{u}}_{t,t-N+1}.\end{aligned} \tag{3.24}$$

However, matrices $\underline{F}_t$ and $\underline{G}_t$ cannot be identified because the separate measurement of the system output $\mathbf{y}$ is not available. In addition, the last approximation in Eq. (3.24) is based on $\overline{\Delta\mathbf{y}}_{t,t-N+1} \approx \overline{\Delta\mathbf{e}}_{t,t-N+1}$ by assuming that the reference increments $\overline{\Delta\mathbf{y}}^{ref}_{t,t-N+1}$ are all zeros. In another word, the reference is assumed constant in the time horizon $[t-N, t]$, which is heavy and not valid in many situations/moments.

Therefore, this section makes a more general assumption as follows:

Assumption 2. *The output reference signal is first-order continuous and can partially be a function of the system output.*

Under this assumption, the dynamics of the reference can be written in a discrete form with a high sampling frequency:

$$\mathbf{y}^{ref}_{t+1} = f^{ref}(\mathbf{y}^{ref}_t, \mathbf{y}_t). \tag{3.25}$$

Taking the Taylor expansion, the incremental form of this discrete reference signal can be written as

$$\Delta \mathbf{y}^{ref}_{t+1} \approx F^{ref}_{t-1} \Delta \mathbf{y}^{ref}_{t} + G^{ref}_{t-1} \Delta \mathbf{y}_{t}, \tag{3.26}$$

where $F^{ref}_{t-1} = \frac{\partial f^{ref}(\mathbf{y}^{ref},\mathbf{y})}{\mathbf{y}^{ref}}|_{\mathbf{y}^{ref}_t} \in \mathscr{R}^{p\times p}$, and $G^{ref}_{t-1} = \frac{\partial f^{ref}(\mathbf{y}^{ref},\mathbf{y})}{\mathbf{y}}|_{\mathbf{y}_t} \in \mathscr{R}^{p\times p}$. In this PO condition, the reference is stochastic, and thus the matrix F^{ref}_{t-1} is time-varying. If the reference is independent of the system, matrix G^{ref}_{t-1} is a zero matrix. In some other cases, matrix G^{ref}_{t-1} is non-zero when the reference is partially determined by the system output, such as moving targets equipped with anti-tracking systems.

The system dynamics in Eqs. (3.6) and (3.22) can be combined with the reference dynamics in Eq. (3.26) and represented as:

$$\begin{bmatrix} \Delta \mathbf{x}_{t+1} \\ \Delta \mathbf{y}^{ref}_{t+1} \end{bmatrix} = \mathscr{F}_{t-1} \begin{bmatrix} \Delta \mathbf{x}_{t} \\ \Delta \mathbf{y}^{ref}_{t} \end{bmatrix} + \mathscr{G}_{t-1} \Delta \mathbf{u}_t, \tag{3.27}$$

$$\Delta \mathbf{e}_{t+1} = \mathscr{H}_t \begin{bmatrix} \Delta \mathbf{x}_{t+1} \\ \Delta \mathbf{y}^{ref}_{t+1} \end{bmatrix}, \tag{3.28}$$

where $\mathscr{F}_{t-1} = \begin{bmatrix} F_{t-1} & 0 \\ G^{ref}_{t-1} H_{t-1} & F^{ref}_{t-1} \end{bmatrix} \in \mathscr{R}^{(n+p)\times(n+p)}$, $\mathscr{G}_{t-1} = \begin{bmatrix} G_{t-1} \\ 0 \end{bmatrix} \in \mathscr{R}^{(n+p)\times m}$, and $\mathscr{H}_t = \begin{bmatrix} H_t & -I \end{bmatrix} \in \mathscr{R}^{p\times(n+p)}$. The new system in Eq. (3.27) consists of the system state and reference dynamics, which are also known as an augmented system [89, 98]. Eq. (3.28) represents the observations in the PO condition.

Therefore, the system state and output reference can be represented by the previous data on the time horizon [t-M, t]:

$$\begin{bmatrix} \Delta \mathbf{x}_{t+1} \\ \Delta \mathbf{y}^{ref}_{t+1} \end{bmatrix} \approx \widetilde{\mathscr{F}}_{t-1,t-M} \cdot \begin{bmatrix} \Delta \mathbf{x}_{t-M+1} \\ \Delta \mathbf{y}^{ref}_{t-M+1} \end{bmatrix} + \mathscr{U}_M \cdot \overline{\Delta \mathbf{u}}_{t,t-M+1}, \tag{3.29}$$

where symbol $\widetilde{\mathscr{F}}_{t-a,t-b} = \prod_{i=t-a}^{t-b} \mathscr{F}_i = \mathscr{F}_{t-a} \cdot \cdots \cdot \mathscr{F}_{t-b}$,

$\overline{\Delta \mathbf{u}}_{t,t-M+1} = \begin{bmatrix} \Delta \mathbf{u}_t \\ \Delta \mathbf{u}_{t-1} \\ \vdots \\ \Delta \mathbf{u}_{t-M+1} \end{bmatrix} \in \mathscr{R}^{mM}$, and

$\mathscr{U}_M = \begin{bmatrix} \mathscr{G}_{t-1} & \mathscr{F}_{t-1}\mathscr{G}_{t-2} & \ldots & \widetilde{\mathscr{F}}_{t-1,t-M+1} \cdot \mathscr{G}_{t-M} \end{bmatrix} \in \mathscr{R}^{(n+p)\times mM}$. Similarly, the tracking error can be represented by previous system state and control input on the time horizon [t-M, t]:

$$\overline{\Delta \mathbf{e}}_{t,t-M+1} \approx \overline{\mathcal{V}}_M \cdot \begin{bmatrix} \Delta \mathbf{x}_{t-M+1} \\ \Delta \mathbf{y}^{ref}_{t-M+1} \end{bmatrix} + \overline{\mathcal{T}}_M \cdot \overline{\Delta \mathbf{u}}_{t,t-M+1}, \tag{3.30}$$

where $\overline{\Delta \mathbf{e}}_{t,t-M+1} = \begin{bmatrix} \Delta \mathbf{e}_t \\ \Delta \mathbf{e}_{t-1} \\ \vdots \\ \Delta \mathbf{e}_{t-M+1} \end{bmatrix} \in \mathscr{R}^{pM}$,

$\overline{\mathscr{V}}_M = \begin{bmatrix} \mathscr{H}_{t-1}\widetilde{\mathscr{F}}_{t-2,t-M} \\ \mathscr{H}_{t-2}\widetilde{\mathscr{F}}_{t-3,t-M} \\ \vdots \\ \mathscr{H}_{t-M} \end{bmatrix} \in \mathscr{R}^{pM\times(n+p)}$ is the observability matrix of this new system, and

$$\overline{\mathscr{T}}_M = \begin{bmatrix} 0 & \mathscr{H}_{t-1}\mathscr{G}_{t-2} & \mathscr{H}_{t-1}\mathscr{F}_{t-2}\mathscr{G}_{t-3} & \cdots & \mathscr{H}_{t-1}\widetilde{\mathscr{F}}_{t-2,t-M+1}\cdot\mathscr{G}_{t-M} \\ 0 & 0 & \mathscr{H}_{t-2}\mathscr{G}_{t-3} & \cdots & \mathscr{H}_{t-2}\widetilde{\mathscr{F}}_{t-3,t-M+1}\cdot\mathscr{G}_{t-M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mathscr{H}_{t-M+1}\cdot\mathscr{G}_{t-M} \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathscr{R}^{pM\times mM}.$$

The necessary condition for a full column rank observability matrix $\overline{\mathscr{V}}_M$ is that M needs to satisfy $M \geq (n+p)/p$. This chapter uses the smallest value for M that meets the condition to make parameters to be identified as less as possible. When the new system is fully observable, $\overline{\mathscr{V}}_M$ has a full column rank, and its left inverse $\overline{\mathscr{V}}_M^{left}$ can be obtained:

$$\overline{\mathscr{V}}_M^{left} = (\overline{\mathscr{V}}_M^T\overline{\mathscr{V}}_M)^{-1}\overline{\mathscr{V}}_M^T. \tag{3.31}$$

Left-multiplying $\overline{\mathscr{V}}_M^{left}$ to Eq. (3.30) and substituting the resulted $\begin{bmatrix} \Delta\mathbf{x}_{t-M+1} \\ \Delta\mathbf{y}_{t-M+1}^{ref} \end{bmatrix}$ into Eq. (3.29) and then the resulted $\begin{bmatrix} \Delta\mathbf{x}_{t+1} \\ \Delta\mathbf{y}_{t+1}^{ref} \end{bmatrix}$ into Eq. (3.28), the dynamics of the measurement can be obtained:

$$\begin{aligned} \Delta\mathbf{e}_{t+1} \approx (&\mathscr{H}_t\mathscr{U}_M - \mathscr{H}_t\widetilde{\mathscr{F}}_{t-1,t-M}\cdot\overline{\mathscr{V}}_M^{left}\overline{\mathscr{T}}_M)\cdot\overline{\Delta\mathbf{u}}_{t,t-M+1} \\ &+ \mathscr{H}_t\widetilde{\mathscr{F}}_{t-1,t-M}\overline{\mathscr{V}}_M^{left}\cdot\overline{\Delta\mathbf{e}}_{t,t-M+1}. \end{aligned} \tag{3.32}$$

This proves that at the moment $\overline{\mathscr{V}}_M$ is full column rank, the tracking error at $(t+1)$ can be represented only by the observations $\mathbf{e}$ and control input $\mathbf{u}$ over the time horizon [t-M, t]:

$$\Delta\mathbf{e}_{t+1} \approx \underline{\mathscr{G}}_t\cdot\overline{\Delta\mathbf{u}}_{t,t-M+1} + \underline{\mathscr{F}}_t\cdot\overline{\Delta\mathbf{e}}_{t,t-M+1}, \tag{3.33}$$

where

$$\underline{\mathscr{G}}_t = (\mathscr{H}_t\mathscr{U}_M - \mathscr{H}_t\widetilde{\mathscr{F}}_{t-1,t-M}\cdot\overline{\mathscr{V}}_M^{left}\overline{\mathscr{T}}_M) \in \mathscr{R}^{p\times Mm}$$

is the extended input distribution matrix, and

$$\underline{\mathscr{F}}_t = \mathscr{H}_t\widetilde{\mathscr{F}}_{t-1,t-M}\overline{\mathscr{V}}_M^{left} \in \mathscr{R}^{p\times Mp}$$

is the extended transition matrix. Matrices $\underline{\mathscr{G}}_t$ and $\underline{\mathscr{F}}_t$ can be identified using LS techniques. With the identified extended incremental model, the one-step prediction of the tracking error can be made:

$$\mathbf{e}_{t+1} = \mathbf{e}_t + \underline{\mathscr{G}}_{t,11}\cdot\Delta\mathbf{u}_t + \underline{\mathscr{G}}_{t,12}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-M+1} + \underline{\mathscr{F}}_t\cdot\overline{\Delta\mathbf{e}}_{t,t-M+1}, \tag{3.34}$$

where $\underline{\mathscr{G}}_{t,11} \in \mathscr{R}^{p\times m}$ and $\underline{\mathscr{G}}_{t,12} \in \mathscr{R}^{p\times(M-1)m}$ are partitioned matrices from $\underline{\mathscr{G}}_t$. Therefore, the control increment $\Delta\mathbf{u}_t$ can be designed based on this prediction and the optimality principle.

This extended incremental model can predict the error one-step ahead without direct reconstruction of the unmeasured internal dynamics and reference dynamics. Therefore, the approximation of the cost function $\widehat{J}^{\mu}$ can be only based on the observed tracking error at the current time $\mathbf{e}_t$ without expanding the dimensionality of the relative state space. Consistently, a systematic cost function approximation is chosen to be quadratic in this tracking error for a kernel matrix $\mathscr{P} \in \mathscr{R}^{p\times p}$ as

$$\widehat{J}^{\mu}(\mathbf{e}_t) = \mathbf{e}_t^T \mathscr{P} \mathbf{e}_t \tag{3.35}$$

so that this optimization problem can have an explicit solution. For continuous tracking problems, the kernel matrix $\mathscr{P}$ shall be a positive definite matrix, and the cost function shall only have one local minimum, which is also the global minimum. Another reason for choosing this cost function is that the one-step cost is a quadratic value in the output tracking error and control input:

$$c(t) = (\mathbf{y}_t - \mathbf{y}_t^{ref})^T Q(\mathbf{y}_t - \mathbf{y}_t^{ref}) + \mathbf{u}_t^T R\mathbf{u}_t, \tag{3.36}$$

where Q and R are positive definite matrices.

The Bellman equation for the true cost-to-go J^{μ} in Eq. (3.8) is rewritten here again:

$$J^{\mu}(t) = \mathbf{e}_t^T Q\mathbf{e}_t + (\mathbf{u}_{t-1} + \Delta\mathbf{u}_t)^T R(\mathbf{u}_{t-1} + \Delta\mathbf{u}_t) + \gamma J^{\mu}(t+1). \tag{3.37}$$

The difference is that $\mathbf{e}_t$ represents the output tracking error in Eq. (3.37). By taking the approximation function in Eq. (3.35) and the one-step prediction in Eq. (3.32), the cost-to-go at the next time instance $J^{\mu}(t+1)$ can be approximated as follows:

$$\begin{aligned}\widehat{J}^{\mu}(t+1) =& \mathbf{e}_{t+1}^T \mathscr{P} \mathbf{e}_{t+1}\\ &\approx [\mathbf{e}_t + \underline{\mathscr{G}}_{t,11}\cdot\Delta\mathbf{u}_t + \underline{\mathscr{G}}_{t,12}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-M+1} + \underline{\mathscr{F}}_t\cdot\overline{\Delta\mathbf{e}}_{t,t-M+1}]^T\cdot\mathscr{P}\\ &\quad\cdot[\mathbf{e}_t + \underline{\mathscr{G}}_{t,11}\cdot\Delta\mathbf{u}_t + \underline{\mathscr{G}}_{t,12}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-M+1} + \underline{\mathscr{F}}_t\cdot\overline{\Delta\mathbf{e}}_{t,t-M+1}].\end{aligned} \tag{3.38}$$

With the optimality principle, the optimal control input can by derived by setting the derivative of the approximated cost function $\widehat{J}^{\mu}$ in Eq. (3.37) w.r.t. $\Delta\mathbf{u}_t$ to zero, as follows:

$$\begin{aligned}\Delta\mathbf{u}_t =& -(R + \gamma\underline{\mathscr{G}}_{t,11}^T \mathscr{P} \underline{\mathscr{G}}_{t,11})^{-1}\\ &\cdot[R\mathbf{u}_{t-1} + \gamma\underline{\mathscr{G}}_{t,11}^T \mathscr{P}(\mathbf{e}_t + \underline{\mathscr{G}}_{t,12}\cdot\overline{\Delta\mathbf{u}}_{t-1,t-M+1} + \underline{\mathscr{F}}_t\cdot\overline{\Delta\mathbf{e}}_{t,t-M+1})].\end{aligned} \tag{3.39}$$

In this equation, $\underline{\mathscr{G}}_t$ and $\underline{\mathscr{F}}_t$ are identified time-varying matrices, which provide the local system model information. The matrix $\mathscr{P}$ provides the cost function information, which can be time-invariant if the policy improved iteratively or be time-varying if the policy updates recursively.

The batch iADP tracking control algorithm with Partial Observability (PO) is given as follows:

Batch iADP-PO algorithm for reference tracking

Evaluation. The cost function kernel matrix $\mathscr{P}$ under current policy can be evaluated and updated recursively for each iteration $j = 0, 1, \ldots$ until convergence:

$$\mathbf{e}_t^T \mathscr{P}^{(j+1)} \mathbf{e}_t = \mathbf{e}_t^T Q \mathbf{e}_t + \mathbf{u}_t^T R \mathbf{u}_t + \gamma \mathbf{e}_{t+1}^T \mathscr{P}^{(j)} \mathbf{e}_{t+1}. \tag{3.40}$$

Policy improvement. The policy improves for the new kernel matrix $\mathscr{P}^{(j+1)}$:

$$\begin{aligned}\Delta\mathbf{u}_t = &-(R + \gamma \underline{\mathscr{G}}_{t,11}^T \mathscr{P}^{(j+1)} \underline{\mathscr{G}}_{t,11})^{-1} \\ &\cdot [R\mathbf{u}_{t-1} + \gamma \underline{\mathscr{G}}_{t,11}^T \mathscr{P}^{(j+1)} (\mathbf{e}_t + \underline{\mathscr{G}}_{t,12} \cdot \overline{\Delta\mathbf{u}}_{t-1,t-M+1} + \underline{\mathscr{F}}_t \cdot \overline{\Delta\mathbf{e}}_{t,t-M+1})].\end{aligned} \tag{3.41}$$

Matrices $\underline{\mathscr{G}}_t$ and $\underline{\mathscr{F}}_t$ are online identified using the LS techniques. The identifiability at each moment relies not only on the full column rank of $\overline{\mathscr{V}}_M$ but also on the excitation. The RLS method can partly relieve the dependence on the data sample at each specific moment.

When the trained policy $\mathscr{P}^{(j)}$ becomes fast and accurate enough, the system does not have much exploration in the relative state space, and consequently, the parameters in $\mathscr{P}^{(j+1)}$ might not be identified using the batch LS method hereafter. Hence, the kernel matrix $\mathscr{P}$ can be continuously updated using RLS with the forgetting factor $\gamma_P^{RLS} = 1$:

Recursive iADP-PO algorithm for reference tracking

Evaluation. The cost function kernel matrix $\mathscr{P}$ within one iteration $j+1$ can be evaluated and updated adaptively for each time step t:

$$\mathbf{e}_t^T \mathscr{P}_t \mathbf{e}_t = \mathbf{e}_t^T Q \mathbf{e}_t + \mathbf{u}_t^T R \mathbf{u}_t + \gamma \mathbf{e}_{t+1}^T \mathscr{P}_{t-1} \mathbf{e}_{t+1}. \tag{3.42}$$

Policy improvement. The policy improves for the new kernel matrix $\mathscr{P}_t$:

$$\begin{aligned}\Delta\mathbf{u}_t = &-(R + \gamma \underline{\mathscr{G}}_{t,11}^T \mathscr{P}_t \underline{\mathscr{G}}_{t,11})^{-1} \\ &\cdot [R\mathbf{u}_{t-1} + \gamma \underline{\mathscr{G}}_{t,11}^T \mathscr{P}_t (\mathbf{e}_t + \underline{\mathscr{G}}_{t,12} \cdot \overline{\Delta\mathbf{u}}_{t-1,t-M+1} + \underline{\mathscr{F}}_t \cdot \overline{\Delta\mathbf{e}}_{t,t-M+1})].\end{aligned} \tag{3.43}$$

The initial value of the kernel matrix for the recursive iADP-PO, $\mathscr{P}_0$, is the last updated $\mathscr{P}^{(j)}$ from the batch iADP-PO. Furthermore, this online recursive adaptation can be applied to changing or faulty systems.

This section proves that the one-step ahead tracking error is predictable in Eqs. (3.23) to (3.33) and elaborates the necessary measurements to make this prediction in Eq. (3.34). Although in this proof the system is augmented as in Eq. (3.27), the derived iADP-PO algorithms do not identify the augmented system or even reconstruct the internal augmented states. Instead, these algorithms only need to identify the extended incremental model with the matrices $\underline{\mathscr{F}}_t$ and $\underline{\mathscr{G}}_t$ and to approximate the true cost-to-go with the cost function kernel matrix $\mathscr{P}$.

3.2.4. INCREMENTAL MODEL ONLINE IDENTIFICATION

Assumption 3. *The (extended) incremental model is identifiable with proper excitation.*

Under the above assumption, this chapter adopts Recursive Least Square (RLS) approaches to online identify the (extended) incremental model.

Incremental Model with Full State Measurements

To implement the iADP-FS algorithms in section 3.2.2, the incremental model needs to be identified with the system transition matrix $\mathbf{F}_{t-1}$ and the input distribution matrix $\mathbf{G}_{t-1}$. The incremental form of the state in Eq. (3.6) can be rewritten row by row as follows:

$$\Delta x_{r,t+1} \approx \begin{bmatrix} \Delta \mathbf{x}_t^T & \Delta \mathbf{u}_t^T \end{bmatrix} \cdot \begin{bmatrix} \mathbf{f}_{r,t-1}^T \\ \mathbf{g}_{r,t-1}^T \end{bmatrix}, \tag{3.44}$$

where $\Delta x_{r,t+1} = x_{r,t+1} - x_{r,t}$ is the increment of rth state element, and $\mathbf{f}_{r,t-1}$ and $\mathbf{g}_{r,t-1}$ are the elements of rth row vector of $\mathbf{F}_{t-1}$ and $\mathbf{G}_{t-1}$. The identification of these parameters with RLS is usually row by row. Since they share the same covariance matrix, they can also be identified together as in the parameter matrix $\Theta_{t-1} = \begin{bmatrix} \mathbf{F}_{t-1}^T \\ \mathbf{G}_{t-1}^T \end{bmatrix} \in \mathscr{R}^{(n+m)\times n}$.

The state prediction equation can be written as

$$\Delta \widehat{\mathbf{x}}_{t+1}^T = X_t^T \widehat{\Theta}_{t-1}, \tag{3.45}$$

where $X_t = \begin{bmatrix} \Delta \mathbf{x}_t \\ \Delta \mathbf{u}_t \end{bmatrix} \in \mathscr{R}^{(n+m)\times 1}$ stands for the input information of the incremental model identification. The RLS approach adopted to identify the incremental model is presented as follows [99]:

$$\epsilon_t = \Delta \mathbf{x}_{t+1}^T - \Delta \widehat{\mathbf{x}}_{t+1}^T, \tag{3.46}$$

$$\widehat{\Theta}_t = \widehat{\Theta}_{t-1} + \frac{Cov_{t-1} X_t}{\gamma_m^{RLS} + X_t^T Cov_{t-1} X_t} \epsilon_t, \tag{3.47}$$

$$Cov_t = \frac{1}{\gamma_m^{RLS}} \left(Cov_{t-1} - \frac{Cov_{t-1} X_t X_t^T Cov_{t-1}}{\gamma_m^{RLS} + X_t^T Cov_{t-1} X_t} \right), \tag{3.48}$$

where $\epsilon_t \in \mathscr{R}^n$ is the prediction error, also called *innovation*, $Cov_t \in \mathscr{R}^{(n+m)\times(n+m)}$ is the estimation covariance matrix, and $\gamma_m^{RLS} \in [0,1]$ is the forgetting factor in the RLS approach for the incremental model identification. For these time-variant model parameters, the forgetting factor is chosen as $\gamma_m^{RLS} = 0.8$. In addition, this chapter initializes the $\mathbf{F}$ matrix as an identity matrix, the $\mathbf{G}$ matrix as a zero matrix, and the covariance matrix as an identity matrix multiplied by a large positive value (1000 in this chapter) $Cov_0 = 1000\mathbf{I}$.

Extended Incremental Model with Partial Observability

In the implementation of the iADP-PO algorithms in section 3.2.3, the extended incremental model needs to be identified with the extended transition matrix $\underline{\mathscr{F}}_t$ and the extended input distribution matrix $\underline{\mathscr{G}}_t$. The incremental form of the observation in Eq. (3.33) can be rewritten row by row as follows:

$$\Delta e_{r,t+1} \approx \begin{bmatrix} \overline{\Delta \mathbf{e}}_{t,t-M+1}^T & \overline{\Delta \mathbf{u}}_{t,t-M+1}^T \end{bmatrix} \cdot \begin{bmatrix} \underline{\mathscr{F}}_{r,t}^T \\ \underline{\mathscr{G}}_{r,t}^T \end{bmatrix}, \tag{3.49}$$

where $\Delta e_{r,t+1} = e_{r,t+1} - e_{r,t}$ is the increment of rth tracking error element, and $\underline{\mathscr{F}}_{r,t}$ and $\underline{\mathscr{G}}_{r,t}$ are the elements of rth row vector of $\underline{\mathscr{F}}_t$ and $\underline{\mathscr{G}}_t$. Similarly, although the identification of these parameters are usually row by row, they can also be identified together as in the parameter matrix $\underline{\Theta}_t = \begin{bmatrix} \underline{\mathscr{F}}_t^T \\ \underline{\mathscr{G}}_t^T \end{bmatrix} \in \mathscr{R}^{(p+m)M\times p}$ because they share the same covariance matrix.

The tracking error prediction equation can be written as

$$\Delta\hat{\mathbf{e}}_{t+1}^T = \overline{X}_t^T \underline{\hat{\Theta}}_t, \tag{3.50}$$

where $\overline{X}_t = \begin{bmatrix} \overline{\Delta\mathbf{e}}_{t,t-M+1} \\ \Delta\mathbf{u}_{t,t-M+1} \end{bmatrix} \in \mathscr{R}^{(p+m)M\times 1}$ stands for the input information of the extended incremental model identification. The RLS algorithm adopted to identify the extended incremental model is presented as follows:

$$\underline{\epsilon}_t = \Delta\mathbf{e}_{t+1}^T - \Delta\hat{\mathbf{e}}_{t+1}^T, \tag{3.51}$$

$$\underline{\hat{\Theta}}_t = \underline{\hat{\Theta}}_{t-1} + \frac{\underline{Cov}_{t-1}\overline{X}_t}{\gamma_{em}^{RLS} + \overline{X}_t^T \underline{Cov}_{t-1}\overline{X}_t}\underline{\epsilon}_t, \tag{3.52}$$

$$\underline{Cov}_t = \frac{1}{\gamma_{em}^{RLS}}\left(\underline{Cov}_{t-1} - \frac{\underline{Cov}_{t-1}\overline{X}_t\overline{X}_t^T\underline{Cov}_{t-1}}{\gamma_{em}^{RLS} + \overline{X}_t^T\underline{Cov}_{t-1}\overline{X}_t}\right), \tag{3.53}$$

where $\underline{\epsilon}_t \in \mathscr{R}^p$ is the innovation term, $\underline{Cov}_t \in \mathscr{R}^{(n+p)M\times(n+p)M}$ is the estimation covariance matrix, and $\gamma_{em}^{RLS} \in [0,1]$ is the forgetting factor in the RLS approach for the extended incremental model identification. For these time-variant model parameters, the forgetting factor is also chosen as $\gamma_{em}^{RLS} = 0.8$. In addition, this chapter initializes the extended transition matrix as $\underline{\mathscr{F}}_0 = [\mathbf{I}\ 0]$, the extended input distribution matrix $\underline{\mathscr{G}}_0$ as a zero matrix, and the covariance matrix also as $\underline{Cov}_0 = 1000\mathbf{I}$.

The RLS approach used in this chapter possesses a significant advantage over the piecewise Ordinary Least Square (OLS) method: RLS has fewer issues with persistent excitation. The OLS method needs to do a matrix inversion at each update [46, 91]. If there is not enough excitation at that moment, the matrix might not be invertible, and the parameters cannot be identified. The RLS method, on the other hand, does not need to do matrix inversions because it uses the matrix inversion lemma to update the covariance matrix, which contains the information of that inverted matrix. Therefore, it can effectively identify parameters of the time-varying system and also keep the parameters stable when the excitation is not enough.

3.3. Tracking Control Simulation

This section firstly introduces an aerospace application for validation of the proposed algorithms. The second part explicates the implementation of the aforementioned algorithms and discusses some related issues, including the persistent excitation and the forgetting factor.

3.3.1. SPACECRAFT WITH LIQUID SLOSHING

In many real applications, there are often internal dynamics that cannot be modeled or directly controlled. One of the most undesired, unknown, and uncertain dynamics involved in the spacecraft attitude control is the liquid sloshing. This phenomenon is caused, for example, by the liquid in the fuel tank sloshing around due to the forces and torques acting on the spacecraft. In turn, the sloshing fuel will interact with the fuel tank and thus perturb the spacecraft. This chapter applies the model-free iADP method to control a satellite disturbed by the liquid sloshing. The model in this section is used as a validation of the proposed method.

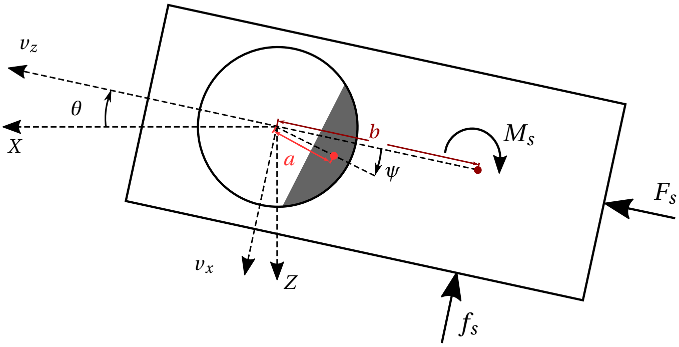


Figure 3.1: A satellite model with pendulum dynamics modeling liquid sloshing.

As seen in Fig. 3.1, apart from the liquid the satellite can be seen as a rigid body, whose mass and moment of inertia are m_s and I_s. The liquid sloshing can be represented by a pendulum [10–13], whose mass, moment of inertia, and angle w.r.t. the spacecraft longitudinal axis are m_p, I_p, and ψ, respectively. The attitude angle θ is defined w.r.t. a fixed reference (X, Z), and v_x and v_z are the axial and transverse components of the velocity of the center of the fuel tank. Three external forces/moments are acting through/on the satellite center of mass: the transverse force f_s and the pitching moment M_s are the satellite attitude control inputs, and the thrust F_s is constant.

Although the proposed method is model-free, the satellite model is included in this chapter for validation and reproduction. The satellite and analogous liquid sloshing dynamic and kinematic state equations are directly taken from [11, 12] and presented as follows:

$$(m_s+m_p)(\dot{v}_x+v_z\dot{\theta})+m_s b\dot{\theta}+m_p a(\ddot{\psi}+\ddot{\theta})sin(\psi)+m_p a(\dot{\psi}+\dot{\theta})^2 cos(\psi)=F_s, \tag{3.54}$$

$$(m_s+m_p)(\dot{v}_z-v_x\dot{\theta})+m_s b\ddot{\theta}+m_p a(\ddot{\psi}+\ddot{\theta})cos(\psi)-m_p a(\dot{\psi}+\dot{\theta})^2 sin(\psi)=f_s, \tag{3.55}$$

$$m_s b(\dot{v}_z-v_x\dot{\theta})+(I_s+m_s b^2)\ddot{\theta}-\kappa\dot{\psi}=M_s+bf_s, \tag{3.56}$$

$$(m_p a^2+I_p)(\ddot{\psi}+\ddot{\theta})+m_p a[(\dot{v}_x+v_z\dot{\theta})sin(\psi)+(\dot{v}_z-v_x\dot{\theta})cos(\psi)]+\kappa\dot{\psi}=0, \tag{3.57}$$

where the parameters used in off-line simulations are
model A: $m_s=600kg$, $I_s=720kg/m^2$, $m_p=100kg$, $I_p=90kg/m^2$, $a=0.3m$, $b=0.3m$, $F_s=500N$, and $\kappa=0.19kg\cdot m^2/s$.

From Eqs. (3.54) and (3.55), the translational movement related variables can be isolated as follows:

$$(\dot{v}_x + v_z\dot{\theta}) = \frac{F_s - m_s b\dot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})sin(\psi) - m_p a(\dot{\psi}+\dot{\theta})^2 cos(\psi)}{m_s + m_p}, \tag{3.58}$$

$$(\dot{v}_z - v_x\dot{\theta}) = \frac{f_s - m_s b\ddot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})cos(\psi) + m_p a(\dot{\psi}+\dot{\theta})^2 sin(\psi)}{m_s + m_p}. \tag{3.59}$$

The rotational variables can be separated from the translational variables by substituting Eqs. (3.58) and (3.59) into Eqs. (3.56) and (3.57):

$$\begin{aligned} &m_s b[f_s - m_s b\ddot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})cos(\psi) + m_p a(\dot{\psi}+\dot{\theta})^2 sin(\psi)] \\ &+(m_s+m_p)(I_s+mb^2)\ddot{\theta} - (m_s+m_p)\kappa\dot{\psi} = (m_s+m_p)(M_s+bf_s), \end{aligned} \tag{3.60}$$

$$\begin{aligned} &m_p a\{sin(\psi)[F_s - m_s b\dot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})sin(\psi) - m_p a(\dot{\psi}+\dot{\theta})^2 cos(\psi)] \\ &+cos(\psi)[f_s - m_s b\ddot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})cos(\psi) + m_p a(\dot{\psi}+\dot{\theta})^2 sin(\psi)]\} \\ &+(m_s+m_p)(m_p a^2 + I_p)(\ddot{\psi}+\ddot{\theta}) + (m_s+m_p)\kappa\dot{\psi} = 0. \end{aligned} \tag{3.61}$$

The rotational motion of the satellite and the pendulum do not contain any translational variables. Hence, the MIMO nonlinear model of the satellite attitude control problem has been generated for simulations.

3.3.2. Implementation Issues

This section discusses some issues related to the implementation of iADP algorithms, including the forgetting factors in the cost function and in RLS approaches, the reliance on persistent excitation, and model-free approaches.

Forgetting Factor

In ADP methods, the forgetting factor γ represents the importance of the upcoming states in the future and is usually chosen as $\gamma \in (0,1)$. Therefore, the infinite sum has a finite value as long as the cost sequence is bounded, and the agent is not "myopic" in being concerned only with maximizing immediate cost [31]. Compared to LADP methods, iADP methods prefer a smaller γ because the nonlinear system, as well as the reference signal in the tracking task, has more uncertainties. In this chapter, $\gamma = 0.5$. Note that the control performance is not very sensitive to γ as $\gamma = 0.2$ and $\gamma = 0.8$ have similar performance to the nominated value, but the magnitude of the value function may increase with a larger γ.

In RLS techniques, the forgetting factor $\gamma^{RLS} \in [0,1]$ represents the reliance on the previous data samples in the past when making an identification. If the parameters are assumed time-invariant, such as the P matrix under a certain policy, the forgetting factor is chosen to be $\gamma_P^{RLS} = 1$. Thus, all the samples are equally taken into account. Otherwise, the forgetting factor is often chosen as $\gamma^{RLS} < 1$, and the most recent data receive higher credit, such as $\gamma_m^{RLS} = 0.8$ in the online identification of time-varying (extended) incremental models.

Persistent Excitation

The trade-off between exploration and exploitation is a common issue in RL methods. In control field, a good evaluation also relies on a proper exploration of the state space, which is referred to as Persistent Excitation (PE). The iADP method is designed based on the optimality principle, which is represented as the exploitation. Therefore, the first purpose of PE is to provide exploration so as to achieve a better evaluation of the current policy. Second, although RLS, compared to OLS, depends less on PE, PE is still imperative for identifying the (extended) incremental model, especially when it is time-varying.

This requirement also explains the need for the off-line batch iADP algorithms. The policy derived from iADP methods is explicitly based on the cost function. If the policy updated recursively at the initial unstable stage, the true cost-to-go also changes in each iteration. Consequently, the approximated cost function is not reliable and the system state may diverge even faster. On the other hand, the off-line learning with batch LS method keeps the policy fixed in each iteration, and the PE helps the exploration of the state space. Therefore, the kernel matrix will converge iteratively.

This chapter introduces an input disturbance, which is a sum of sinusoidal signals. This disturbance persistently excites the system for identification of the system and exploration of the state space. On the other hand, disturbances are usually undesirable inputs in the real world. Therefore, the control task is to track the reference signal as well as to reject the disturbance, which also increases the difficulty of the task.

Model-free Approaches and Partial Observability

Although the rotational motion of the satellite model with liquid sloshing can be analogous to the pendulum dynamics as Eqs. (3.60) and (3.61) in the previous section. The iADP model-free nonlinear method does not need any a priori knowledge about the model, but only assumes a general model, which can represent any structure and parameters:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t) + \mathbf{d}(t)], \tag{3.62}$$

where $\mathbf{x} = [\theta, \dot{\theta}, \psi, \dot{\psi}]^T$ is the state vector, $\mathbf{u} = [f_s, M_s]^T$ is the control input vector of the system since the thrust F_s is constant, and $\mathbf{d}$ is the input disturbance vector. The batch iADP-FS algorithm for reference tracking in Eqs. (3.16) and (3.17) can be used to update the matrix P and train the policy off-line. The system information are provided by the incremental model identified online using Eqs. (3.46) to (3.48). After the averaged one-step-cost is smaller than a threshold, the kernel matrix will keep updating using the recursive iADP-FS algorithm in Eqs. (3.18) and (3.19).

In aerospace applications, it is common that the only measurement is the output tracking error, such as the relative position and the relative angle pointing to the target. This chapter also applies iADP-PO algorithms to this spacecraft attitude control problem in such situations. The only observation is the output tracking error $\mathbf{e} = \mathbf{y} - \mathbf{y}^{ref}$, where $\mathbf{y} = [\theta, \psi]^T$. For regulation problems, the observation becomes the system output $\mathbf{y}$, and the system is fully observable with input/output measurements. This chapter will apply an unknown time-varying reference signal, which can be seen as a continuous-time stochastic process. It brings the stochastic property to the observations. Although the system output cannot be separated from the observations, the output tracking error is still predictable as proved in section 3.2.3. This problem can be solved with the iADP-PO algorithms in Eqs. (3.40) to (3.43).

3.4. Results and Discussion

This chapter demonstrates the performance of the proposed iADP-FS and iADP-PO algorithms in different observability conditions by simulation experiments on satellite attitude control problems disturbed by liquid sloshing. In both observability conditions, the policy is trained off-line using the before-mentioned *model A*, and is then applied to a different reference and/or a different model with changed parameters as *model B*: $m_s = 1200kg$, $I_s = 900kg/m^2$, $m_p = 50kg$, $I_p = 80kg/m^2$, $a = 0.2m$, $b = 0.5m$, $F_s = 600N$, and $\kappa = 0.19kg \cdot m^2/s$.

3.4.1. iADP with Full State Measurements for Tracking Control

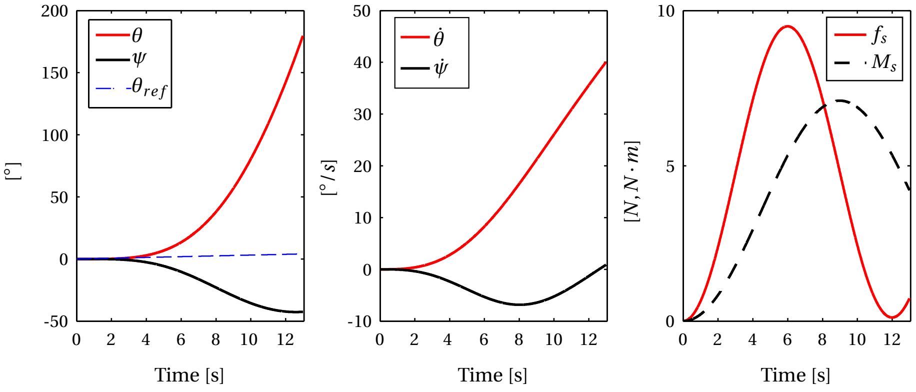


Figure 3.2: The control performance on *model A*, using the initial policy (1st iteration) with PE.

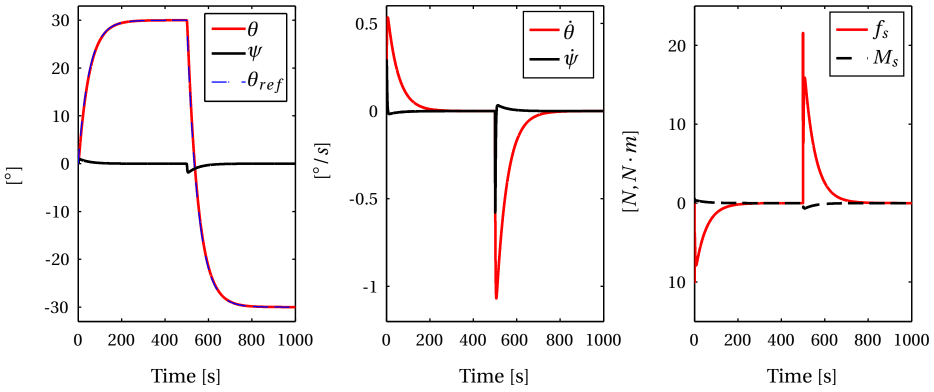


Figure 3.3: The control performance on *model A*, using the trained policy (5th iteration) to track a filtered doublet signal.

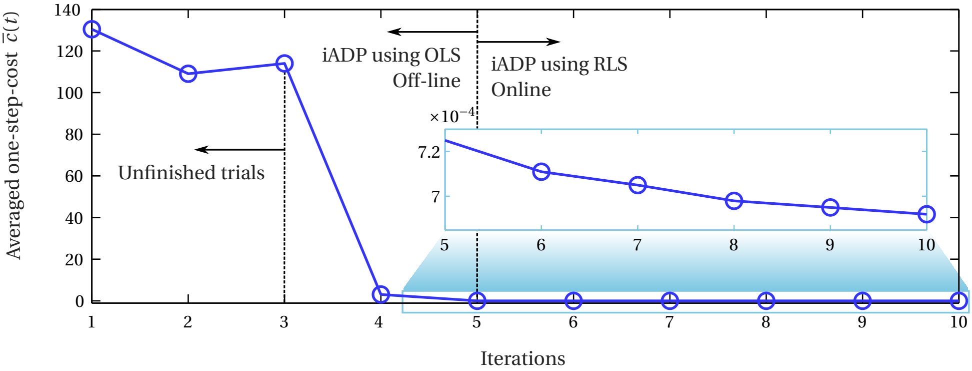


Figure 3.4: The averaged one-step-cost during the training on *model A* in tracking the filtered doublet signal.

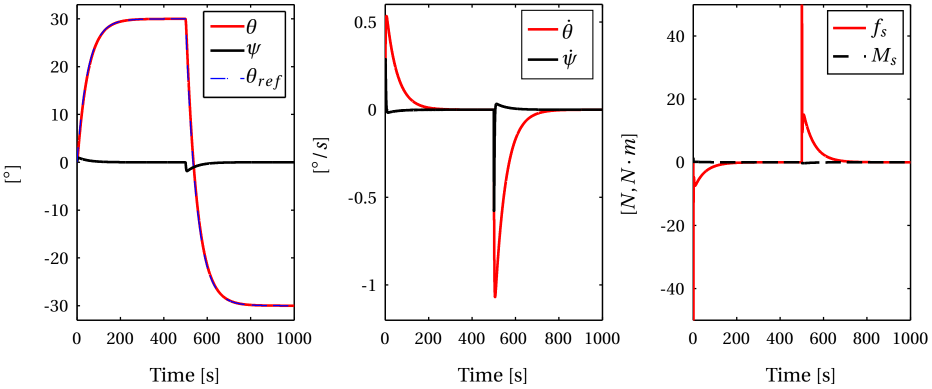


Figure 3.5: The online control performance on a different model (*model B*), using the recursive iADP-FS algorithm to track the filtered doublet signal.

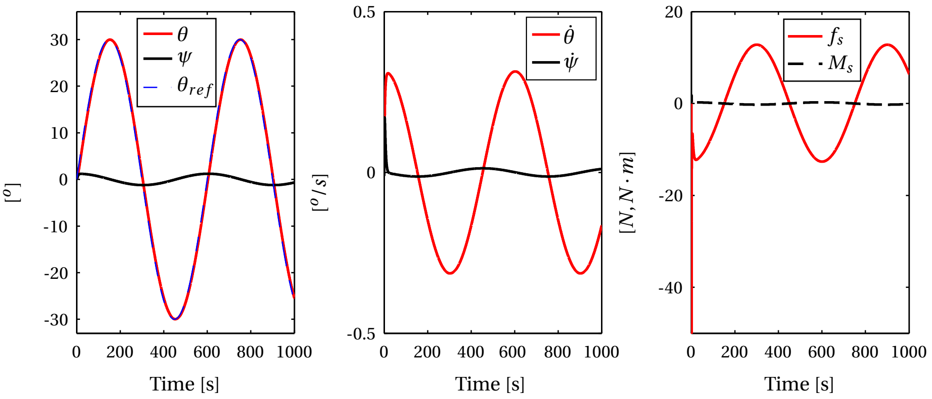


Figure 3.6: The online control performance on a different model (*model B*), using the recursive iADP-FS algorithm to track a different reference (a sinusoidal signal).

The iADP-FS algorithms are based on the assumption that the full states (θ, ψ, $\dot{\theta}$, and $\dot{\psi}$) and the control input (f_s, M_s) are measurable. The target is to track a reference signal of θ while stabilizing ψ. It often happens in the initial attitude acquisition or tracking a (relatively) moving target.

This iADP method starts with an off-line learning stage of 10 iterations. In this stage, the termination of each iteration can happen 1) when it finishes the whole trial and reaches the time limit 1000 seconds or 2) when any of the states reaches their limit: 180° for θ and ψ and $50°/s$ for $\dot{\theta}$ and $\dot{\psi}$. The initial kernel matrix P^0 is a zero matrix. Therefore, the first iteration is an open loop system with sinusoidal PE, as shown in Fig. 3.2. This iteration stops at around $13s$ when θ reaches 180°.

The first tracking task is to follow a filtered doublet signal in the attitude angle. As illustrated in Fig. 3.3, the control performance of the trained policy in the 5th iteration can track the reference of θ, stabilize ψ, and reject the input disturbance. Figure 3.4 shows the averaged one-step-cost in 10 iterations of the off-line stage. After the 5th iteration, the averaged one-step-cost is below the threshold, and the adaptation of the policy uses the RLS approach afterward. The first 3 iterations cannot finish the whole trial and stop because they reach the state limit.

This chapter also examines the situations that the target system is different from the model that is used to train the policy, which often happens in practice. To validate the proposed control method, the policy trained with $model A$ will be applied to $model\ B$ using the recursive iADP-FS online. As depicted in Fig. 3.5, the recursive iADP algorithm can be applied to a different model starting with the trained policy without loss of accuracy. The main reason is that the iADP method separates the cost function with an approximation and the model information with the online identified incremental model. When the system changes, the value function does not necessarily change a lot, while the incremental model will change immediately using the RLS approach.

Except for the system model, the reference dynamics may also affect the control performance. To further validate the online adaptability of the proposed method, the policy trained off-line with $model\ A$ and doublet reference in Fig. 3.3 is applied to a different model, $model\ B$, as well as a different reference: a sinusoidal signal. As visible in Fig. 3.6, the recursive iADP-FS algorithm can update the incremental model and the policy adaptively and consequently track the dynamical reference of θ, minimize ψ, and also reject the input disturbance.

3.4.2. iADP with Partial Observability for Tracking Control

In this section, the proposed iADP-PO algorithms are applied to the same problem with only the measurement of the control input $\mathbf{u}$ and output tracking error $\mathbf{e} = \mathbf{y} - \mathbf{y}^{ref}$, where $\mathbf{y} = [\theta, \psi]^T$. The reference θ^{ref} may have its own stochastic dynamics, and the separate measurement of θ or θ^{ref} is not available. Since the reference ψ^{ref} is constant, the extended dynamical states in Eqs. (3.27) and (3.28) are (θ, ψ, $\dot{\theta}$, $\dot{\psi}$, and θ^{ref}), which are 5 in total, and the order of matrices become $\mathscr{F}_{t-1} \in \mathscr{R}^{5\times5}$, $\mathscr{G}_{t-1} \in \mathscr{R}^{5\times2}$, and $\mathscr{H}_t \in \mathscr{R}^{2\times5}$. Consequently, the extended observability matrix in Eq. (3.29) becomes $\overline{\mathscr{V}}_M \in \mathscr{R}^{2M\times5}$. In this case, to make $\overline{\mathscr{V}}_M$ full column rank, the number of data samples M in Eqs. (3.41) and (3.43) is chosen to be $M = 3$. Note that, as long as the system is observable, the necessary condition $M \geq (n+p)/p$ mentioned in section 3.2.3 still holds for those cases that only

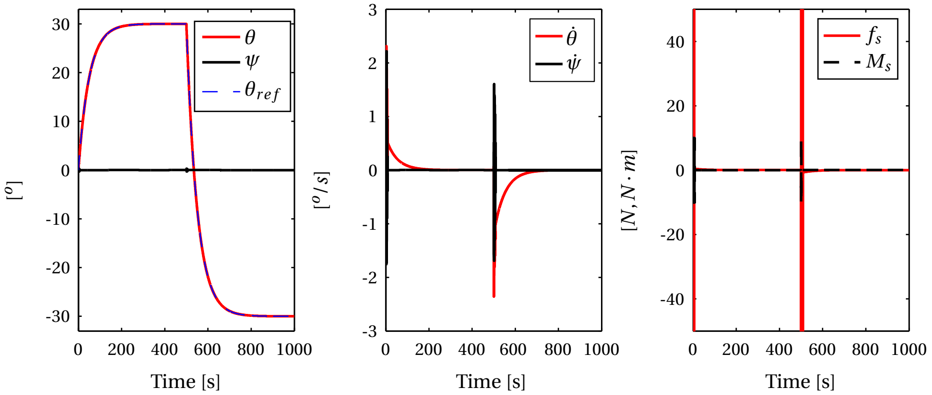


Figure 3.7: The control performance on *model A*, using the trained policy (9th iteration) to track a filtered doublet signal.

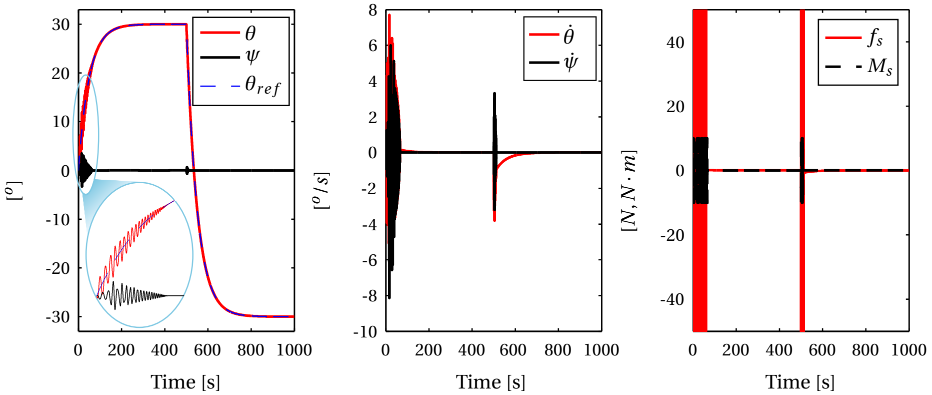


Figure 3.8: The online control performance on a different model (*model B*), using the recursive iADP-PO algorithm to track the filtered doublet signal.

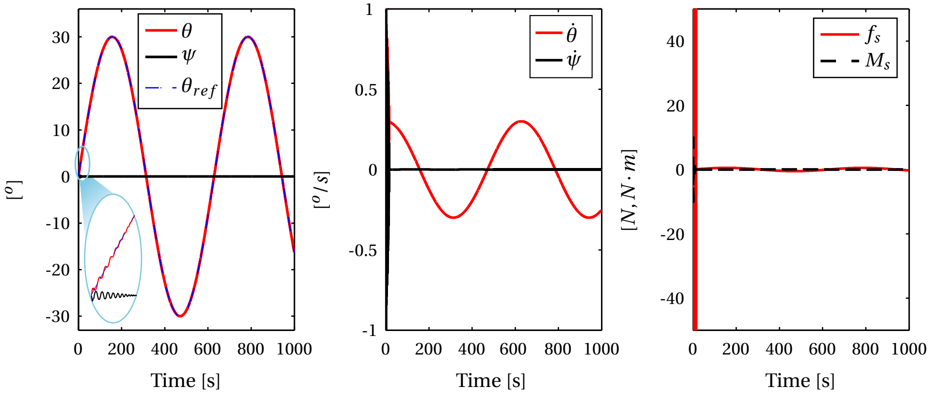


Figure 3.9: The online control performance on a different model (*model B*), using the recursive iADP-PO algorithm to track a different reference (a sinusoidal signal).

part of the output references is dynamical. Therefore, when the iADP-PO algorithms are applied to an tracking control application, no matter how many dynamical output references there are, $M \geq (n+p)/p$ can be used as the selection criteria.

Figure 3.7 illustrates the control performance of the policy trained off-line with *model A* using the batch iADP-PO algorithm. Note that the angular rates $\dot{\theta}$, $\dot{\psi}$ and even the attitude angle θ are not measurable in the PO cases. As seen in sub-figure (a), the control performance in terms of the attitude tracking is at the same level of accuracy as the full state feedback control in Fig. 3.3 (a). However, in sub-figures (b) and (c), there are oscillations in angular rates and control input at the beginning and at $500s$. These differences are due to the partial observability of the system.

The optimization in iADP algorithms is based on the cost function approximation and the online identified (extended) incremental model. In the PO condition, the one-step-cost, as well as the approximated cost function, are lack of internal states information, which is the angular rate in this case. Hence, the optimization does not take these internal measurements into account, and consequently, the policy does not regulate them and may induce oscillations. This is, to some extent, inevitable in PO cases without a priori information of the system.

Another reason for the oscillation is the delay and inaccuracy of the extended incremental model identification. The incremental model identification is based on the measurements and the assumption of the first-order Taylor series expansion. Hence, the first-order derivation calculated from the discrete measurement often has a smaller absolute value than the true value and will lead to an over-high control input design. This phenomenon is worse, when the full observation is not available, because the extended incremental model is identified with the measurements in a few time steps in the past, and this introduces a time delay and expands the time span of the measurements for the local model identification. In the PO condition, the extended incremental model needs more time to be identified. In addition, since the system output θ and the reference θ^{ref} are combined in the measurement, the sudden changes in the reference dynamics will deteriorate the online identification at the moment the reference signal is not 1st order continuous.

To validate the online adaptability of the recursive iADP-PO algorithm, the policy trained off-line with *model A*, and doublet reference is applied to a different model (*model B*), as seen in Fig. 3.8, and to a different model (*model B*) as well as for a different reference (a sinusoidal signal), as shown in Fig. 3.9. Comparing the Figs. 3.8 and 3.9, the iADP algorithms under the PO condition performs better when tracking a 1st order continuous reference as assumed in section 3.2.3 . As seen in these figures, the oscillation happens 1) at the beginning of each iteration on *model B*, and 2) when the reference suddenly changed. The reason is that both the mismatched model and the suddenly changed reference affect the identification of the extended incremental model.

By taking a closer look at the first a few seconds in Figs. 3.8 and 3.9, when the policy trained on one model applied to another model, the system presents more oscillations compared to the full state feedback control performance. This phenomenon is also obvious when comparing Fig. 3.8 to Fig. 3.7. Except that the system needs more time to identify the extended incremental model, another reason is that the extended kernel matrix $\mathscr{P}$ contains less information and hence may need more time for online adaptation when applied to a different model. After this online adaptation of $\mathscr{P}$ and a proper identified extended incremental model, the tracking control performance becomes satisfactory.

3.5. CONCLUSION

This chapter proposes an adaptive nonlinear control method for optimal tracking based on incremental Approximate Dynamic Programming (iADP). Nonlinear ADP methods often use highly nonlinear functions to approximate the true cost-to-go and the system model. The iADP method, on the other hand, uses an (extended) incremental model to deal with the nonlinearity of the unknown system and uncertainty of the environment. Therefore, this method still can apply a systematic quadratic cost function to generate an explicit optimal control algorithm, which simplifies the design process of nonlinear ADP methods. This method does not need any a priori knowledge of the system dynamics, online identification of the global model nor even an assumption of the time scale separation, but only an online identified (extended) incremental model.

This study expands the iADP method to tracking problems for MIMO nonlinear systems and to partial observable control problems. First, when the direct measurement of the full state is available, the incremental model can be identified and used to predict the next state. With this prediction, the optimal control increment can be designed to track the current reference signal. Second, when the only measurement is the output tracking error involved with stochastic dynamical reference, the system becomes partially observable. Under the assumption of the first-order continuity, the next output tracking error is proved to be predictable with the partial observations over a long enough time horizon and an online identified extended incremental model. Because iADP methods have a separate structure to represent the local system dynamics, the cost function can be less dependent on the system or the reference and only need to be a rough approximation of the cost-to-go. This approximation is a quadratic function only of the current tracking error without expanding the dimension of the state space for the cost function to an augmented one.

For each observability condition, two algorithms are developed for off-line batch learning and online recursive adaptation, respectively. The off-line learning algorithms use OLS to improve the policy iteratively. After the averaged one-step-cost is below a threshold, the online algorithms will be applied to update the policy recursively at each time step. The recursive iADP algorithms can also be used online in real systems, which are mismatched from the original model and are not easy to reset after failure. These algorithms are applied to an attitude control problem of a simulated satellite disturbed by liquid sloshing. The results show that the off-line trained policy in either observability condition rejects the disturbance and tracks references accurately. When the trained policy is applied to a different system and/or a different reference, the control performance reaches the same level of accuracy in a few seconds. The online learning using the recursive iADP ensures the adaptability of the proposed method.

The performance of the tracking control using the proposed algorithms is observed to be very promising, especially for unknown nonlinear systems with only partial observability. The quadratic cost function simplifies the approximation of the cost-to-go and also provides an explicit optimal control solution. Although beneficial for most tracking control problems with approximately convex cost-to-go, the quadratic function may not be suitable for systems and tasks with highly nonlinear cost-to-go. For more complex systems and tasks, general nonlinear function approximators, such as piecewise quadratic functions, will be studied in the future. In addition, the iADP-PO algorithms provide insights for future research using only relative states in the aerospace applications/areas, such as air missile guidance, UAV swarm flight control, and aircraft formation flying.

II

Online Adaptive Critic Designs

4

Incremental Model Based Heuristic Dynamic Programming

In the previous chapters, incremental Approximate Dynamic Programming (iADP) methods were proposed to deal with nonlinear systems, while keeping the design process simple and mathematically explicit with a quadratic cost function. As these methods are suitable only for control problems with approximately convex true cost-to-go, this chapter proposes a new approach for more general nonlinear problems, called Incremental model based Heuristic Dynamic Programming (IHDP). This chapter starts with a conventional Heuristic Dynamic Programming (HDP) algorithm using a nonlinear global system model, in Section 4.2. After that, the development of the IHDP algorithm is presented in Section 4.3. Section 4.4 validates the proposed method in several nonlinear control problems. The results show that the IHDP method can speed up online learning and can improve the control performance compared to the conventional HDP method. Moreover, it is adaptive and robust to internal uncertainties and external disturbances.

This chapter is based on the following articles:
Y. Zhou, E. van Kampen, and Q. P. Chu. Incremental model based heuristic dynamic programming for nonlinear adaptive flight control. in International Micro Air Vehicle Conference and Competition 2016 (IMAV 2016) (Beijing, PR of China, 2016) p. 173-180 [91].
Y. Zhou, E. van Kampen, and Q. P. Chu. Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming. in 68th International Astronautical Congress (IAC) (Adelaide, Australia, 2017) [100].

A self-learning controller which makes quick and successful adaptations to new conditions can considerably benefit autonomous operations of launch vehicles. To provide a model-free, adaptive process for optimal control, approximate dynamic programming has been introduced to aerospace engineering. A widely used structure of approximate dynamic programming for nonlinear systems is heuristic dynamic programming. This chapter proposes a new method using an incremental model in heuristic dynamic programming to improve the online learning capacity. This method generates an adaptive near-optimal controller online without a priori knowledge of the system dynamics or off-line learning of the system model. This chapter validates the proposed method on two different online tracking problems: missile flight control and satellite attitude control disturbed by liquid sloshing. The results demonstrate that the incremental model based heuristic dynamic programming method accelerates online learning, improves the precision, and can deal with a wider range of initial states compared to the conventional heuristic dynamic programming method, and robust to internal uncertainties and external disturbance.

4.1. Introduction

Spaceflight is one of the greatest technological achievements of humankind. It is used in space exploration, launching communication satellites and space telescopes, and also to make observations from a different perspective of the phenomena that occur on Earth. Adaptive and robust control takes a vital part in these activities. Most existing control methods are based on an accurate model and may require a thorough evaluation and test. In addition, the system uncertainties, such as model mismatch, aerodynamic uncertainties, internal dynamics and external disturbances, may also degrade the control performance of model-based control methods. Therefore, this chapter aims to develop a nonlinear self-adaptive flight control algorithm for a wide range of spaceflight.

Reinforcement Learning (RL) methods have been introduced to solve nonlinear, optimal control problems without using accurate models [29, 31]. These methods link bio-inspired artificial intelligence techniques to the field of control in order to overcome some of the limitations and challenges of control methods that require accurate models. In the control field, RL is also referred to as Approximate Dynamic Programming (ADP), which solves nonlinear, optimal, fault-tolerant control problems with large or continuous state spaces [50, 53, 101–103]. Different from traditional RL methods, ADP applies a function approximator to approximate the value/cost of the states, theoretically to any arbitrary degree of precision. ADP methods exploit this function to approximate the utility of any state and to obtain optimal solutions of the Hamilton-Jacobi-Bellman (HJB) equations. With this approximate function, ADP methods can tackle the ‘curse of dimensionality’, which traditional RL methods are confronted with [32, 34].

Adaptive Critic Designs (ACDs, also known as Actor-Critic designs, ACs) constitute a class of ADP methods that separates the evaluation and improvement using parametric structures [50]. The most basic and widely used form is Heuristic Dynamic Programming (HDP). It consists of an actor, a critic, and an approximate plant structure connecting the actor and the critic [32, 34, 50, 104]. An alternative approach is Action Dependent Heuristic Dynamic Programming (ADHDP), which does not need a plant approximation, but directly connects the output of the actor to the input of the critic. From a theoretical point of view, the actor output is not necessarily an input to the critic, which estimates the value/cost function; and from a practical perspective, the extra input can increase the complexity of the critic. Furthermore, an investigation on the difference between HDP and ADHDP found that HDP controller, compared with the ADHDP controller, can operate in a wider range of flight conditions, adapts quicker to changed plant dynamics, and has a higher success ratio in controlling an F-16 aircraft model [54]. Therefore, this chapter only considers HDP instead of its AD form.

HDP methods can control highly nonlinear systems by exploiting nonlinear function approximators, such as Artificial Neural Networks (ANNs). With these approximators, they can identify the system dynamics globally and generate the control laws adaptively. However, online identification of the global model is not a trivial task, especially when the system is complex and highly nonlinear [8, 9, 23–27]. It needs a certain time to approximate a feasible model with a nonlinear approximator. HDP method may even need an off-line identification process beforehand using representative simulation models which are also difficult to obtain. Furthermore, it requires adequate computing power, which flight control systems often lack, to perform the approximation of the desired system dynamics.

System nonlinearity can also be dealt with by means of the incremental control technique without identifying the global system [63, 65–67]. This technique has been successfully applied to design adaptive controllers, such as Incremental Nonlinear Dynamic Inversion (INDI) [63, 65, 66] and Incremental Backstepping (IBS) [67]. These methods are based on a linear time-varying approximation of the original system when assuming a sufficiently high sample rate for discretization. However, these methods have not addressed optimization or synthesis of designed closed-loop systems. Incremental Approximate Dynamic Programming (iADP) was proposed for the first time to control unknown nonlinear systems. This approach uses a quadratic function to approximate the cost function [46, 80]. Therefore, it is suitable only for systems with approximately convex cost functions.

This chapter develops a model-free adaptive control approach for unknown nonlinear systems to improve the online learning capability: the Incremental model based Heuristic Dynamic Programming (IHDP). It uses linearized incremental models of the original nonlinear system instead of the artificial neural networks that are often used in HDP controllers. This method is called model-free because it does not need any a priori information of the system dynamics at the beginning of the algorithm nor online identification of the global nonlinear system, but only the online identified incremental model. This algorithm can be seen as an extension of the iADP algorithms developed in [46, 80] with a more general cost function approximator.

The remainder of this chapter is structured as follows. Section 4.2 starts with an introduction of the traditional HDP method and then describes the most widely used approximator: ANN. Section 4.3 proposes the IHDP method using the incremental approach and discusses some implementation related issues. Then, section 4.4 applies the proposed method to online tracking control problems on two different systems and quantitatively presents how much the IHDP method can improve the online performance. Lastly, section 4.5 shows the advantages and disadvantages of using the incremental approach with HDP and addresses the challenges and possibilities of the future research.

4.2. Foundations

4.2.1. HDP Framework and Global System Model

Heuristic Dynamic Programming (HDP) algorithms, similar to other ADP methods, operate by alternating between two steps: policy evaluation, implemented by the critic, and policy improvement, implemented by the actor [41, 53, 55, 105]. Fig.4.1 shows a schematic diagram of an HDP controller, which uses three artificial neural networks to approximate the actor, the critic, and the global system model with weights $\mathbf{w}_a$, $\mathbf{w}_c$, and $\mathbf{w}_m$, respectively.

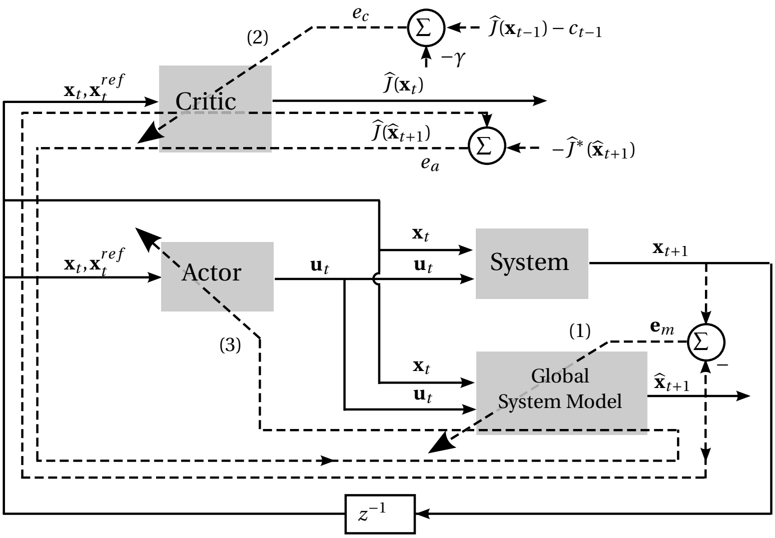


Figure 4.1: A schematic diagram of HDP using a global system model, where solid lines represent the feedforward flow of signals, and dashed lines represent the BP pathways.

The global system model approximates the system dynamics and outputs the estimated next state, $\hat{\mathbf{x}}_{t+1} \in \mathscr{R}^n$. The inputs of the network are the current state, $\mathbf{x}_t \in \mathscr{R}^n$, and the system input, $\mathbf{u}_t \in \mathscr{R}^m$. The adaptation algorithms of both the critic and the actor are based on this system model. The system model updates its weights by minimizing the difference between the measured state $\mathbf{x}_t$ and the estimated state $\hat{\mathbf{x}}_t$:

$$E_m(t) = \frac{1}{2}\mathbf{e}_m^2(t), \tag{4.1}$$

where

$$\mathbf{e}_m(t) = \mathbf{x}_t - \hat{\mathbf{x}}_t. \tag{4.2}$$

The system model weights are updated according to the gradient-descent algorithm with a learning rate η_m:

$$\mathbf{w}_m(t+1) = \mathbf{w}_m(t) + \Delta\mathbf{w}_m(t), \tag{4.3}$$

where

$$\begin{aligned}\Delta\mathbf{w}_m(t) &= -\eta_m\cdot\frac{\partial E_m(t+1)}{\partial\mathbf{w}_m(t)}\\ &= -\eta_m\cdot\frac{\partial E_m(t+1)}{\partial\widehat{\mathbf{x}}_{t+1}}\frac{\partial\widehat{\mathbf{x}}_{t+1}}{\partial\mathbf{w}_m(t)}.\end{aligned} \tag{4.4}$$

4.2.2. ANN AND BACK-PROPAGATION

The actor, critic, and global model are approximated with ANNs, or more specifically Multilayer Perceptrons (MLP), which consist of multiple, fully connected, and feedforward layers of nodes. Each network has an input layer, a hidden layer, and an output layer. Each node in the hidden layer is a neuron with a continuous, nonlinear hyperbolic tangent activation function σ:

$$\sigma(o) = \frac{1-e^{-o}}{1+e^{-o}}. \tag{4.5}$$

Its derivative is continuous and positive at every point:

$$\frac{\partial\sigma(o)}{\partial o} = \frac{1}{2}(1-\sigma(o)^2). \tag{4.6}$$

In fully connected multilayer neural networks, the input of the next layer consists of the outputs of this layer and sometimes also a bias term. Because the output of a hyperbolic tangent function is bounded with $(-1,1)$, and the output of the hidden layer is multiplied with an output weight, the neural network with bias terms can theoretically approximate any value. The detailed neural network calculation and Back-Propagation (BP) algorithms have been provided in [91] and will not be carried out in this chapter.

The actor and critic networks in both HDP and IHDP methods have the same setting in this chapter. The number of hidden layer neurons in the actor and the critic is 6, and in the global system network is 10. The network weights are updated recursively using the Least Mean Square (LMS) method. The neural network weights are limited within the range $[-30,30]$ to prevent their sudden growth to infinity.

4.3. INCREMENTAL MODEL BASED HEURISTIC DYNAMIC PROGRAMMING

This section proposes an adaptive controller for nonlinear systems without a priori knowledge of the system dynamics, namely Incremental model based Heuristic Dynamic Programming (IHDP). This method is devised based on the HDP method and incremental control techniques. As illustrated in Fig. 4.2, the IHDP controller uses only two artificial neural networks to approximate the actor and the critic with weights w_a and w_c, and an incremental model to find the system dynamics at the current moment. Therefore, the design of the IHDP controller relieves the off-line learning stage. This section will devise and elaborate this new method using the incremental approach.

4.3.1. IHDP FRAMEWORK AND ADAPTATION RULES

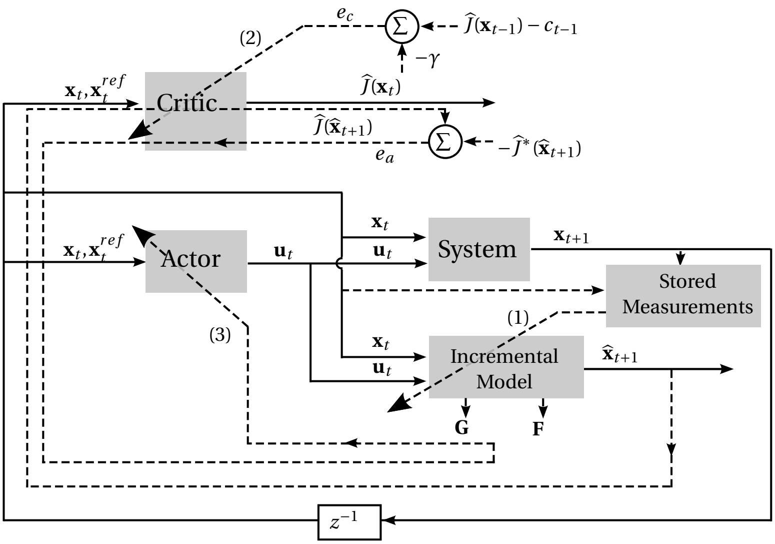


Figure 4.2: A schematic diagram of IHDP using a time-varying incremental model, where solid lines represent the feedforward flow of signals, and dashed lines represent the adaptation pathways.

Incremental Model

Incremental techniques identify the incremental model at the current time by using the conditions of the system in an instant before [46, 65]. This technique can be applied to nonlinear systems with sufficiently high sample rate. The dynamic and kinematic equations of a nonlinear system, such as an aircraft, can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t)], \tag{4.7}$$

where $\mathbf{x} \in \mathscr{R}^n$ is the system state vector, $\mathbf{u} \in \mathscr{R}^m$ is the control input vector, and $f[\mathbf{x}(t), \mathbf{u}(t)] \in \mathscr{R}^n$ provides the physical evaluation of the state vector over time.

The system dynamics around the condition of the system at time t_0 can be linearized approximately by using the first-order Taylor series expansion:

$$\begin{aligned} \dot{\mathbf{x}}(t) \approx \dot{\mathbf{x}}(t_0) &+ F[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{x}(t) - \mathbf{x}(t_0)] \\ &+ G[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{u}(t) - \mathbf{u}(t_0)], \end{aligned} \tag{4.8}$$

where $F[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{x}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)} \in \mathscr{R}^{n\times n}$ is the system matrix of the linearized model at time t_0, and $G[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{u}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)} \in \mathscr{R}^{n\times m}$ is the control effectiveness matrix.

The states and state derivatives of the system are assumed to be measurable, which means $\Delta\dot{\mathbf{x}}(t), \Delta\mathbf{x}(t), \Delta\mathbf{u}(t)$ are measurable. Therefore, the model around time t_0 can be written in an incremental form:

$$\Delta\dot{\mathbf{x}}(t) \simeq F[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{x}(t) + G[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{u}(t). \tag{4.9}$$

The current incremental model can be identified with a Least Squares (LS) technique. This model can be used to obtain an approximated value of $\frac{\partial \mathbf{x}(t+1)}{\partial \mathbf{u}(t)}$ without using an ANN model.

Most of the physical systems are continuous, while measurements are often discrete. With a constant, high data sampling frequency, the nonlinear system can be written in a discrete form as follows:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{4.10}$$

where $f(\mathbf{x}_t, \mathbf{u}_t) \in \mathscr{R}^n$ provides the system dynamics.

When the sample time Δt is sufficiently small, the system dynamics around $\mathbf{x}_t$ can be linearized by taking the Taylor expansion, as follows:

$$\mathbf{x}_{t+1} \approx \mathbf{x}_t + \mathbf{F}_{t-1} \cdot (\mathbf{x}_t - \mathbf{x}_{t-1}) + \mathbf{G}_{t-1} \cdot (\mathbf{u}_t - \mathbf{u}_{t-1}), \tag{4.11}$$

where $\mathbf{F}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times n}$ is the system transition matrix, and $\mathbf{G}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times m}$ is the input distribution matrix at time step $t-1$ for discretized systems. The incremental form of this discrete nonlinear system can be obtained as follows:

$$\Delta\mathbf{x}_{t+1} \approx F_{t-1}\Delta\mathbf{x}_t + G_{t-1}\Delta\mathbf{u}_t, \tag{4.12}$$

With the high-frequency sample data and the relatively slow-variant system assumption, the current linearized model can be identified from L previous measurements using LS methods. With the identified $\widehat{F}_{t-1}$ and $\widehat{G}_{t-1}$ matrix, the next state can be predicted as follows:

$$\widehat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \widehat{F}_{t-1}\Delta\mathbf{x}_t + \widehat{G}_{t-1}\Delta\mathbf{u}_t. \tag{4.13}$$

THE CRITIC

The critic network approximates the true cost-to-go $J(\mathbf{x}_t)$, which is the cumulative sum of future costs c_l from any initial state $\mathbf{x}_t$:

$$J^{\mu}(\mathbf{x}_t) = \sum_{l=t}^{\infty} \gamma^{l-t} c_l, \tag{4.14}$$

where μ is the current policy (control law), which is provided by the actor, and $\gamma \in (0,1)$ is a scalar called *discount factor* or *forgetting factor*. The discount factor ensures that the cost for any state is finite and provides a reasonable evaluation and approximation to infinite-horizon problems as well as problems involving a finite but very large number of stages. By adjusting γ, it is able to control the extent to which the short-term cost or long-term cost is concerned [105].

The cost c_l at a future time l is often a function of the state at that time, and called the *one-step cost function*. In this chapter, it is defined quadratically, so as to minimize the cost of the system state $\mathbf{x}_l$ approaching the reference signal $\mathbf{x}_l^{ref}$, as follows:

$$c_l = c(\mathbf{x}_l, \mathbf{x}_l^{ref}) = (\mathbf{x}_l - \mathbf{x}_l^{ref})^T Q(\mathbf{x}_l - \mathbf{x}_l^{ref}), \tag{4.15}$$

where $Q \in \mathscr{R}^{n\times n}$ is a positive definite matrix. To normalize the effect of each state, we usually use normalization factors in the Q matrix: $Q = diag\{\zeta_1, \zeta_2, ..., \zeta_n\}$, where ζ is a given weight to indicate the importance of the cost for the related state approaching the reference.

HDP methods are on-policy Temporal Difference (TD) methods. They iteratively update the critic by estimating the true cost-to-go for the current policy and update the actor to change the policy towards greediness [31]. The target for the critic update is $c_{t-1} + \gamma \widehat{J}(\mathbf{x}_t)$. Thus, the evaluation of the critic is the TD error:

$$e_c(t) = c_{t-1} + \gamma \widehat{J}(\mathbf{x}_t) - \widehat{J}(\mathbf{x}_{t-1}), \tag{4.16}$$

where $\widehat{J}(\mathbf{x}_t)$ is the approximated cost function from state $\mathbf{x}_t$ under the current policy. Note that $\widehat{J}$ is a function of $\mathbf{x}_t$ and $\mathbf{w}_c(t)$ and is represented by a static neural network.

The critic network is updated so as to minimize the defined error function for the critic $E_c(t)$:

$$E_c(t) = \frac{1}{2} e_c^2(t). \tag{4.17}$$

The weights of the critic network are updated according to the gradient-descent algorithm with a learning rate η_c:

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \Delta \mathbf{w}_c(t), \tag{4.18}$$

where

$$\begin{aligned} \Delta \mathbf{w}_c(t) &= -\eta_c \cdot \frac{\partial E_c(t)}{\partial \mathbf{w}_c(t)} \\ &= -\eta_c \cdot \frac{\partial E_c(t)}{\partial \widehat{J}(\mathbf{x}_t)} \cdot \frac{\partial \widehat{J}(\mathbf{x}_t)}{\partial \mathbf{w}_c(t)}. \end{aligned} \tag{4.19}$$

With a converged policy, the critic and its network parameters $\mathbf{w}_c(t)$ will also converge.

The Actor

The actor is used to find the policy which minimizes the defined cost function $\widehat{J}(\mathbf{x}_t)$. Therefore, the target for the actor update is $J^*(t) = 0$, and the error function for the actor, $E_a(t)$, is defined as follows:

$$E_a(t) = \frac{1}{2} e_a^2(t), \tag{4.20}$$

$$e_a(t) = \widehat{J}(\mathbf{x}_t) - J^*(t). \tag{4.21}$$

The actor update is complicated because it involves the critic and the model. As illustrated in Figs. 4.1 and 4.2, the actor weights affect the true cost-to-go $J(\mathbf{x}_{t+1})$ through affecting $\mathbf{x}_{t+1}$ and $\mathbf{u}_t$ along 3rd back-propagation direction. Therefore, the actor network weights can be updated according to the gradient-descent algorithm with a learning rate η_a:

$$\mathbf{w}_a(t+1) = \mathbf{w}_a(t) + \Delta \mathbf{w}_a(t), \tag{4.22}$$

where

$$\begin{aligned} \Delta \mathbf{w}_a(t) &= -\eta_a \cdot \frac{\partial E_a(t+1)}{\partial \mathbf{w}_a(t)} \\ &= -\eta_a \cdot \frac{\partial E_a(t+1)}{\partial \widehat{J}(\mathbf{x}_{t+1})} \frac{\partial \widehat{J}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}} \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t} \frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}. \end{aligned} \tag{4.23}$$

The approximated model can output the estimation of the next state $\widehat{\mathbf{x}}_{t+1}$ for an input $\mathbf{u}_t$. This also helps to get the useful term $\frac{\partial \widehat{\mathbf{x}}(t+1)}{\partial \mathbf{u}_t}$, which approximates $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$, in updating the actor network [54]. Therefore, Eq. (4.23) can be rewritten as follows:

$$\Delta \mathbf{w}_a(t) = -\eta_a \frac{\partial E_a(t+1)}{\partial \widehat{J}(\widehat{\mathbf{x}}_{t+1})} \frac{\partial \widehat{J}(\widehat{\mathbf{x}}_{t+1})}{\partial \widehat{\mathbf{x}}_{t+1}} \frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}(t)} \frac{\partial \mathbf{u}(t)}{\partial \mathbf{w}_a(t)}. \tag{4.24}$$

The structure of the actor network in IHDP is the same as the one in HDP controller. However, the actor network update is easier and faster than the one in HDP controller, since it involves an incremental model instead of the global system model. The input distribution matrix G of the incremental model can be online identified and be directly used to approximate the derivative of the next state w.r.t. the system input, $\frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t}$:

$$\frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t} \approx \widehat{G}_{t-1}. \tag{4.25}$$

Therefore, with a direct online identification of the incremental model, this method simplifies the approach of updating the actor network weights and accelerates the learning.

4.3.2. Incremental Model Online Identification

The system transition matrix F_{t-1} and the input distribution matrix G_{t-1} of the current linearized model are identifiable by using the LS method:

$$\begin{aligned} \Delta x_{r,t-l+1} &= \mathbf{f}_r \Delta \mathbf{x}_{t-l} + \mathbf{g}_r \Delta \mathbf{u}_{t-l} \\ &= \begin{bmatrix} \Delta \mathbf{x}_{t-l}^T & \Delta \mathbf{u}_{t-l}^T \end{bmatrix} \begin{bmatrix} \mathbf{f}_r^T \\ \mathbf{g}_r^T \end{bmatrix}, \end{aligned} \tag{4.26}$$

where $\Delta x_{r,t-l+1} = x_{r,t-l+1} - x_{r,t-l}$ is the increment of rth state element, $\mathbf{f}_r$ and $\mathbf{g}_r$ are the elements of rth row vector of F_{t-1} and G_{t-1}, $l = 1, 2 ... L$ denotes at which time the historic information is available, and L is the data window length. Because there are $n+m$ parameters in the rth row, L needs to satisfy $L \geq (n+m)$. With a piecewise moving window Least Squares method, the linearized system dynamics (rth row) can be identified from L different data points:

$$\begin{bmatrix} \widehat{\mathbf{f}}_r^T \\ \widehat{\mathbf{g}}_r^T \end{bmatrix} = (\mathbf{A}_{t,L}^T \mathbf{A}_{t,L})^{-1} \mathbf{A}_{t,L}^T \mathbf{x}_{r,t,L}, \tag{4.27}$$

where

$$\mathbf{A}_{t,L} = \begin{bmatrix} \Delta \mathbf{x}_{t-1}^T & \Delta \mathbf{u}_{t-1}^T \\ \vdots & \vdots \\ \Delta \mathbf{x}_{t-L}^T & \Delta \mathbf{u}_{t-L}^T \end{bmatrix}, \mathbf{x}_{r,t,L} = \begin{bmatrix} \Delta x_{r,t} \\ \vdots \\ \Delta x_{r,t-L+1} \end{bmatrix}. \tag{4.28}$$

As incremental models are able to describe the system dynamics within a small time range, the window length L is also important. The identified model may not represent the locally linearized model of the nonlinear system with an L larger than necessary. However, if L is too small, it may run into an identifiability problem, especially when the system excitation is not sufficient. Therefore, to choose L depends not only on the sampling frequency and the nonlinearity but also on the intensity of the excitation. This chapter adopts L to be $2 \cdot (n+m)$, which works well in practice.

4.3.3. Implementation Issues

This section addresses some issues for implementing the aforementioned algorithms, including the excitation of the system, the structured actor network, and the learning rate.

Persistent Excitation

To accomplish the reference tracking task, an adaptive controller with the actor needs to be found out by minimizing the cost-to-go $J(\mathbf{x}_t)$ with a feasible critic and model. As with other ADP methods, good evaluation depends heavily on the exploration of the state space, which is represented by Persistent Excitation (PE). PE is also imperative for identifying the incremental model. Many different input techniques can be used to excite aircraft modes, such as doublets, 3211 doublets, pseudo-random noise, and classical sine waves.

This chapter introduces an input disturbance, which is a sum of sinusoidal signals. This disturbance persistently excites the system for identification of the system and exploration of the state space in HDP methods. On the other hand, disturbances are usually undesirable inputs in the real world. Therefore, the flight control task is to track the reference signal as well as to stabilize the system, if there is any disturbance. Because of the online learning capability of HDP methods, the disturbance can be compensated without being identified.

Cascaded Actor Network

To take advantages of the physical properties of the air vehicle system, a structured cascaded actor network [53, 54] is used as shown in Fig. 4.3. This cascaded structure separates the inner loop and outer loop control, which provides specific relationships between the angular rate and the attitude. As long as the concerned full states and control input of this air vehicle are known, this structure can be easily implemented in the actor network.

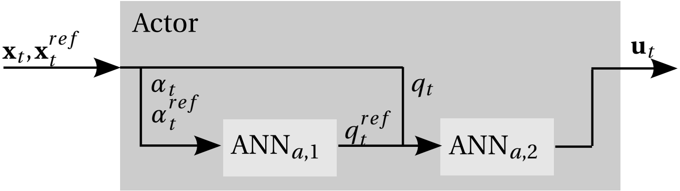


Figure 4.3: The architecture of the cascaded actor network.

Adaptive Learning Rate

HDP methods, similar to other ACDs, are online learning methods in the sense of the adaptation of the critic and the actor. They iteratively learn the critic and the actor, of which value depends on each other. The online learning stage of HDP uses LMS techniques, which update the weights based only on the error at the current time. This method incrementally updates the neural network weights along the steepest descent with a learning rate η. The convergence to the optimal weights depends heavily on a properly chosen η. If η is chosen to be too small, the time to converge will be very large, and it may be trapped in a local optimum. If η is too large, the weight may change by a large amount and oscillate around the optimal weights.

To make the convergence less sensitive to the chosen learning rate, this chapter uses an adaptive learning rate η_t at time t. It is self-tuned in each update. A proper learning rate needs to meet the condition that the new weights decrease the network error. In each time-step, the initial learning rate is assigned by the previous time-step. This method calculates the direction of steepest descent $\frac{\partial E(t)}{\partial \mathbf{w}(t)}$ and searches along this line by halving the learning rate η_t until it meets the condition. With this η_t, the neural network weight can be updated as follows:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t \cdot \frac{\partial E(t)}{\partial \mathbf{w}(t)}. \tag{4.29}$$

If the network errors of the new weights and the old weights have different signs, this new learning rate will also be assigned as the initial learning rate for the next step. Otherwise, the learning rate for the next time-step will be doubled. The initial learning rates $\eta_{0,0}$ at time $t = 0$ for the model, the critic, and the actor are chosen to be 10, 20, and 10, respectively. They are set the same values for the traditional HDP and the IHDP algorithms.

4.4. Numerical Experiments and Results

This section validates the proposed IHDP method to online tracking control problems on two different systems. First, this method is applied to a short period flight control problem for a missile model. This model is simple while nonlinear, which is suitable for an online performance comparison with the conventional HDP method. The second task is a spacecraft attitude control, which is disturbed by internal liquid sloshing. This is for validation on complex Multiple-Input Multiple-Output (MIMO) nonlinear control problems, which contain uncertainties in both the internal system and the external environment.

4.4.1. Missile Flight Control: Comparison between HDP and IHDP

Missile Model

The proposed IHDP method is first applied to a simplified missile model for a comparison with the conventional HDP method. The missile model of a short period flight control problem consists of two states: angle of attack α and pitch rate q. Only the pitch is controlled using elevator deflection δ_e. This nonlinear model in the pitch axis is simulated around a steady wings-level flight condition:

$$\dot{\alpha} = q + \frac{\bar{q}S}{m_a V_T} C_z(\alpha, q, M_a, \delta_e), \tag{4.30}$$

$$\dot{q} = \frac{\bar{q}Sd_l}{I_{yy}} C_m(\alpha, q, M_a, \delta_e), \tag{4.31}$$

where $\bar{q}$ is dynamic pressure, S is reference area, m_a is mass, V_T is speed, d_l is reference length, I_{yy} is pitching moment of inertia, C_z is the force coefficient in body Z-direction, and C_m is the pitch moment coefficient. C_z and C_m are nonlinear functions of angle of attack α, pitch rate q, Mach number M_a and elevator deflection δ_e. In this simulation experiment, an air vehicle model [68, 69, 81] is taken in the pitch plane for $-10^o < \alpha < 10^o$, and the Mach number M_a is set to be 2.2.

There are several reasons for using this missile model: 1) This model is simple while nonlinear, which is suitable for a validation of the newly-developed online adaptive IHDP method. 2) For more complex models, conventional HDP may need off-line learning stage [50–53], which will impede a fair comparison to the IHDP method. 3) The missile model can operate at a high and rapidly changing angle of attack, which leads to a high nonlinearity. 4) It is a second-order continuous model, which means it is relatively real and complete. Although the model is only valid within the given flight envelope, the model can be still used out of the envelope numerically in this simulation.

Although the missile model is presented above, the IHDP model-free method does not need any a priori knowledge about the model, but only assumes a general dynamic and kinematic state equations as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t) + \mathbf{d}(t)], \tag{4.32}$$

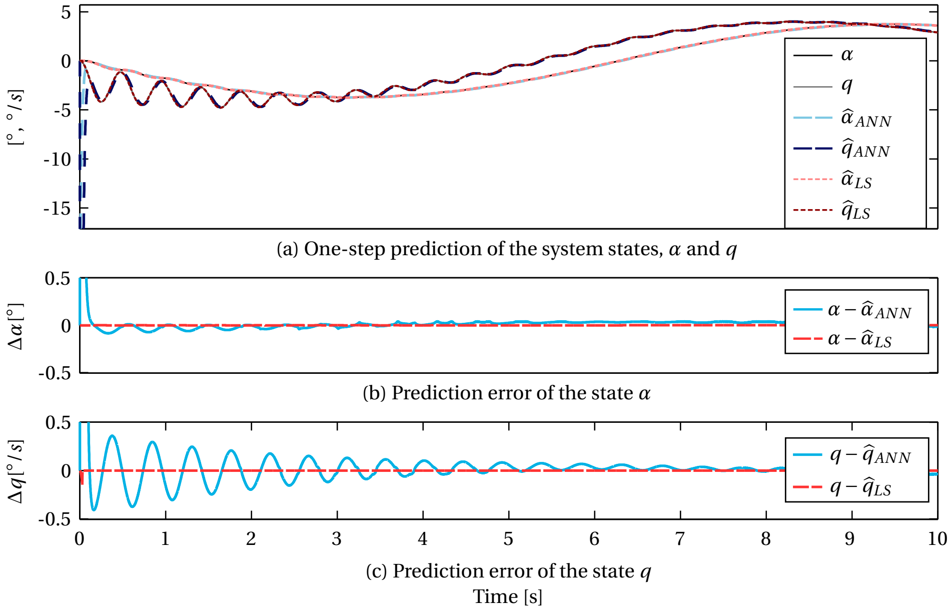
where $\mathbf{x} = [\alpha, q]^T$ is the state vector, $\mathbf{u} = \delta_e$ is the control input vector of the system, and $\mathbf{d}(t)$ is the external disturbance and is set to be caused by the input noise.

Results and Discussions

To solve this tracking problem, two algorithms are applied: 1) traditional HDP uses an artificial neural network to approximate the global system model, and 2) IHDP uses the incremental model to approximate the linearized model at the current moment. In both cases, the identified models have two functions. First, they are used to predict the next states $\widehat{\mathbf{x}}_{t+1}$ so as to estimate the cost of the next state $\widehat{J}(\widehat{\mathbf{x}}_{t+1})$ and its difference from the minimal cost J^*. Second, some information of the models is necessary to estimate the input distribution matrix G at the current moment, which is used to update the actor during the error back-propagation.

Figure 4.4 shows the one-step prediction of α and q using the ANN model and the incremental model identified with LS, respectively, when there is a sinusoidal input excitation. As shown in Fig. 4.4(a), the one-step state predictions are feasible using both the ANN model and the incremental model. However, the online identification using the ANN needs more time to generate an accurate prediction at the beginning. The incremental model, on the other hand, predicts the next state accurately after only a few measurements. Figure 4.4(b) and (c) take a close look at the prediction errors, and show that the prediction using incremental approach has higher precision than using the ANN model. This also explains the reason for having the off-line stage to train the global model in traditional HDP methods.

To back-propagate the error for the actor update, the partial derivative of the next state of the system w.r.t. the input signal, $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$, needs to be approximated. This term is the input distribution matrix for a linearized model with discrete measurements. Figure 4.5 presents the identification results of this term, represented as the matrix

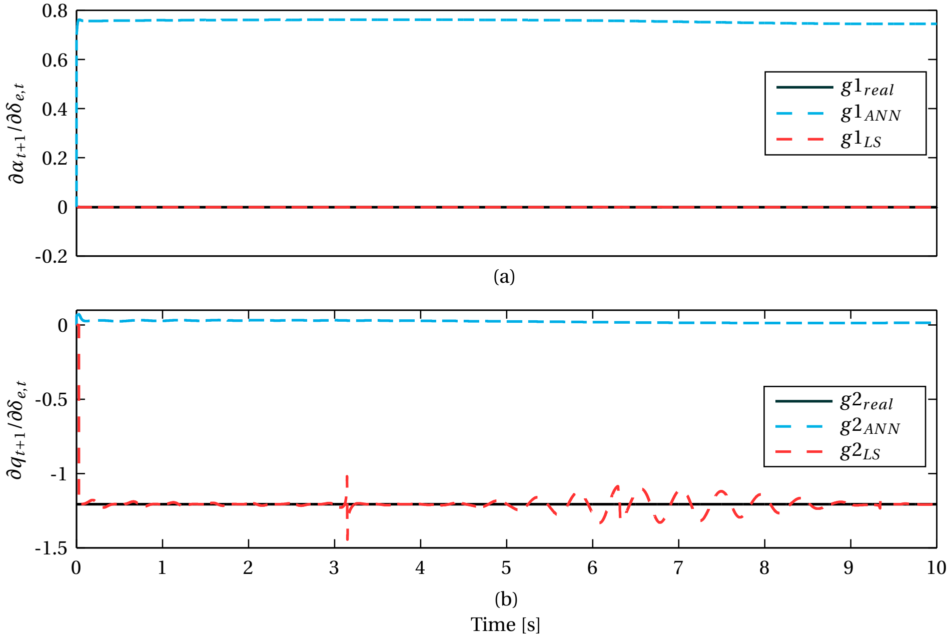


(a) One-step prediction of the system states, α and q

(b) Prediction error of the state α

(c) Prediction error of the state q

Figure 4.4: The one-step state prediction with a sinusoidal input excitation using the online identified ANN and the LS technique.



(a)

(b)

Figure 4.5: Online identified control derivatives, $G = [g1, g2]^T$, using the online identified ANN and the LS technique.

$G = [g1, g2]^T$, using the ANN model and the incremental model, respectively. The matrix identified with the incremental model has higher accuracy than the ANN model.

The reason is that the traditional HDP method updates the model network by minimizing the difference between the measurement of the state and its prediction. The identified ANN model does not have to be a feasible approximation of the derivatives of the system states because it does not build an explicit representation for the derivatives. The IHDP method, on the other hand, identifies the system transition matrix and the input distribution matrix for the linearized model, which directly approximate the control derivatives of the current system states.

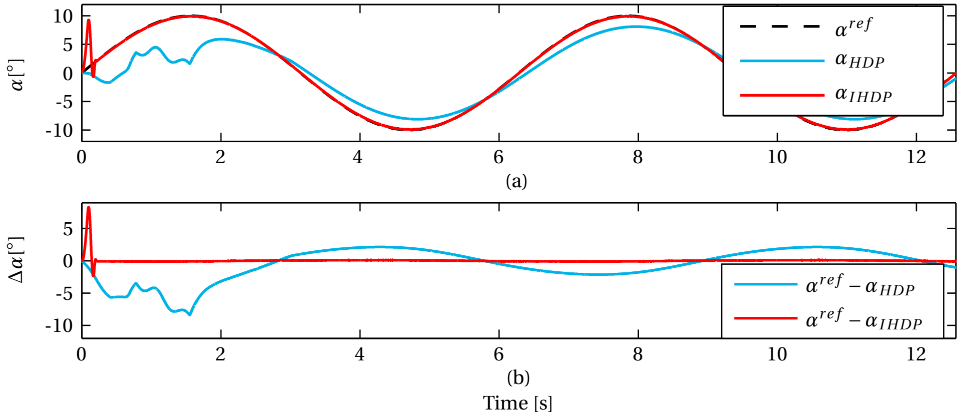
Figures 4.6 to 4.8 compare the performance of the traditional HDP method and IHDP method when applied to an online tracking problem. The sub-figures on the top present how these algorithms control the angle of attack α to track the reference signal, which is a sine function of time with the amplitude of 10°. These simulations run 2 periods of the reference signal, which is 4π seconds. The sub-figures on the bottom provides the error using these two algorithms during this tracking task. Figure 4.6 shows the simulation results when the initial states is zero.

The influence of the initial states is also examined in this research. When the initial state α_0 is ±2°, both methods perform similar to the simulation results with zero initial states, as presented in Figs. 4.7 and 4.8. However, when the initial state α_0 is changed further to ±4° or more, the traditional HDP method cannot track the reference signal. The IHDP method, on the other hand, can deal with different initial states within $[-8°, 8°]$ and performs as good as zero initial states, as shown in Fig. 4.9.

These results uniformly show that, compared to traditional HDP method, IHDP method can identify the local model and reject the disturbance faster, follow the reference signal more precisely, and deal with a wider range of initial states. In realistic cases, the slow online training of the model network may lead to a large overshoot and lost control at the initial stage. Therefore, the traditional HDP usually needs an off-line training of the model before online training of the controller to prevent failures. On the other hand, IHDP does not need off-line learning, because the linearized local model can be identified with only a few measurements. The incremental model identification is faster and more accurate than the neural network model.

Except for the initial conditions of the air vehicle, the initial weights of neural networks can also affect the HDP learning, especially when they are learning online. Therefore, different, random initial neural network weights and different initial states are examined for the traditional HDP and IHDP methods. To reduce the impact of initial weights, the neural network weights are initialized with small, random numbers from the range $(-0.01, 0.01)$. With these initial weights, both methods have a high success ratio: 70.2% for traditional HDP and 86.7% for IHDP, when the initial state $\alpha_0 \in [-2°, 2°]$. Note that this chapter admits a successful trail if it converges within 4π seconds and the error between the target state and the reference signal is no more than 2.5° hereafter, as seen in Figs. 4.6 to 4.8.

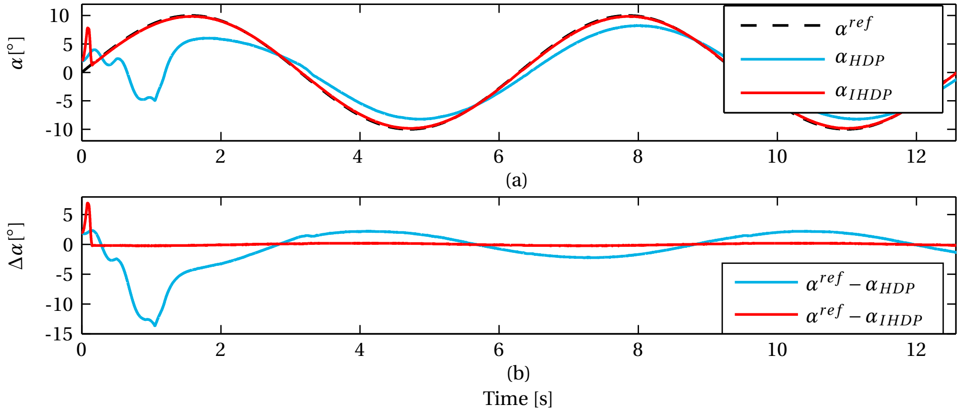
Figure 4.10 presents the changes of the actor output surface during the online tracking problem. This figure examines the actor outputs δ_e of M even distributed state vectors within the sub state space: $\alpha \in [-10, 10]°$, $q \in [-10, 10]°/s$. The output surface at time $t = 0$ is nearly flat, as shown in Fig. 4.10(a), because the initial weights are very



(a)

(b)

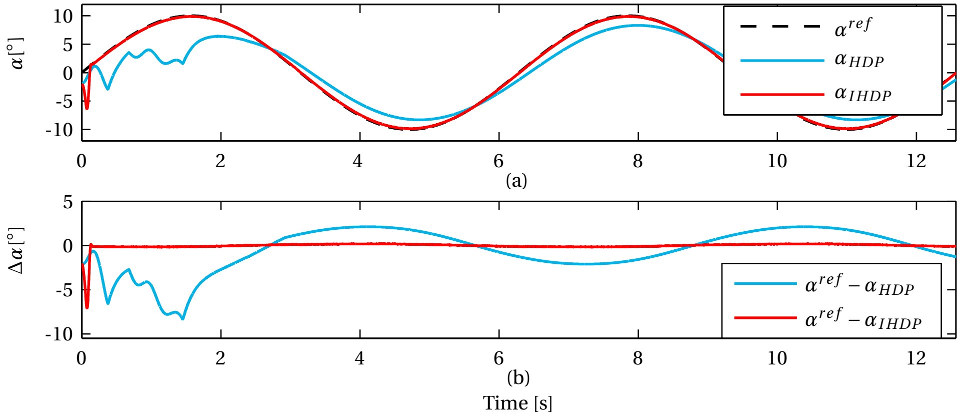
Figure 4.6: Online tracking control with zero initial states using HDP and IHDP approaches.



(a)

(b)

Figure 4.7: Online tracking control with a positive initial state, $\alpha_0 = 2°$, using HDP and IHDP approaches.



(a)

(b)

Figure 4.8: Online tracking control with a negative initial state, $\alpha_0 = -2°$, using HDP and IHDP approaches.

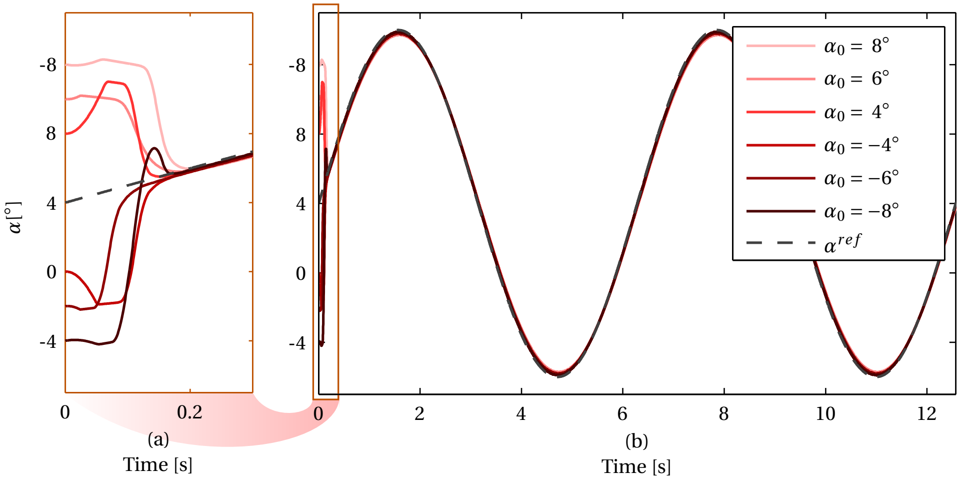
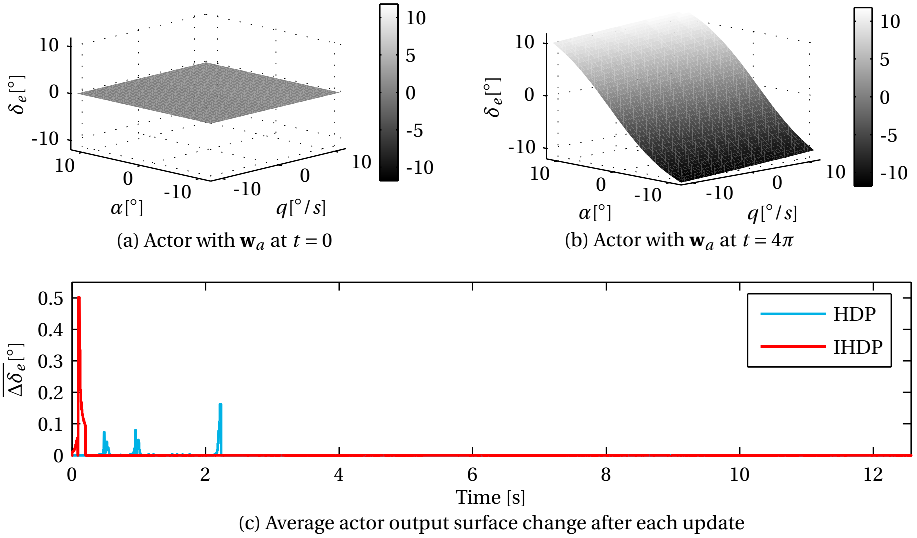


Figure 4.9: Online tracking control with different initial states using the IHDP approach.



(a) Actor with $\mathbf{w}_a$ at $t = 0$

(b) Actor with $\mathbf{w}_a$ at $t = 4\pi$

(c) Average actor output surface change after each update

Figure 4.10: The actor output surface changes during an online tracking problem using HDP and IHDP approaches.

small. Figure 4.10(b) exhibits the output surface after this tracking task at time $t = 4\pi$ (2 periods of the reference signal). The average output surface change $\overline{\Delta\delta_e}$ after each update time t is calculated over the selected state vectors $\mathbf{x}_l$, $l = 1, ..., M$, as follows:

$$\overline{\Delta\delta_e}(t) = \frac{\sum_{l=1}^{M} |\delta_e(\mathbf{x}_l, \mathbf{w}_a(t)) - \delta_e(\mathbf{x}_l, \mathbf{w}_a(t-1))|}{M}. \tag{4.33}$$

Figure 4.10(c) gives the average output surface changes of the actor using traditional HDP and IHDP methods, which indicate the changes of the actor network weights. The differences between traditional HDP and IHDP methods are highlighted in this figure. The traditional HDP updates the actor network later then IHDP because it needs to first have a feasible system model network and critic network. If the system model network converges into a local optima, the actor also adapts to that model, which leads to 3 major changes shown in this figure. On the other hand, IHDP can update the actor as long as it has a feasible critic network and an incremental model, which can be identified immediately.

Table 4.1 illustrates the average tracking error and the average settling time for those successful trails. The average tracking error is calculated within (2π, 4π) seconds. In this chapter, the average settling time is defined as the time that the error reaches and remains within 2.5°. The data in this table is quite revealing in two ways. First, the IHDP method spends less time (0.14 seconds) to find a feasible actor, compared to the traditional HDP (2.11 seconds), which is also apparent in Fig. 4.10. Second, the average tracking error after 2π seconds is decreased from 1.47 degrees with the traditional HDP to 0.11 degrees with the IHDP method, which means that the online IHDP method follows the reference signal more precisely.

Table 4.1: Performance of HDP and IHDP in the Missile Control Simulations

	HDP	IHDP
Average Tracking Error	1.47 °	0.11 °
Average Settling Time	2.11 s	0.14 s

4

4.4.2. SPACECRAFT ATTITUDE CONTROL: VALIDATION OF IHDP WITH UNCERTAINTIES

SPACECRAFT WITH LIQUID SLOSHING

To validate the IHDP method for more complex systems and control tasks, this chapter also carries out simulation experiments on a spacecraft attitude control problem disturbed with liquid sloshing. Liquid sloshing is one of the most undesired, unknown, and uncertain internal dynamics involved in the spacecraft attitude control [10–12]. Although been studied for years, an accurate liquid sloshing model is extremely difficult to obtain [12, 13]. This chapter, therefore, applies the model-free IHDP method to control a satellite disturbed by the liquid sloshing.

Apart from the liquid, the satellite can be seen as a rigid body, whose mass and moment of inertia are m_s and I_s. The liquid sloshing can be represented by a pendulum

[10–13, 84], whose mass, moment of inertia, angle w.r.t. the spacecraft longitudinal axis, and length are m_p, I_p, ψ, and a, respectively. The pendulum point of attachment is in front of the spacecraft center of mass with a distance b. The attitude angle θ is defined w.r.t. a fixed reference, and v_x and v_z are the axial and transverse components of the velocity of the fuel tank center. Three external forces/moments are acting through/on the satellite center of mass: the transverse force f_s and the pitching moment M_s are the satellite attitude control inputs, and the thrust F_s is constant.

With the satellite and analogous liquid sloshing dynamic and kinematic state equations from [11, 12], the rotational variables can be separated from the translational variables as follows [84]:

$$\begin{aligned}&m_s b[f_s - m_s b\ddot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})cos(\psi) + m_p a(\dot{\psi}+\dot{\theta})^2 sin(\psi)]\\&+(m_s+m_p)(I_s+mb^2)\ddot{\theta} - (m_s+m_p)\kappa\dot{\psi} = (m_s+m_p)(M_s+bf_s),\end{aligned} \tag{4.34}$$

$$\begin{aligned}&m_p a\{sin(\psi)[F_s - m_s b\dot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})sin(\psi) - m_p a(\dot{\psi}+\dot{\theta})^2 cos(\psi)]\\&+cos(\psi)[f_s - m_s b\ddot{\theta} - m_p a(\ddot{\psi}+\ddot{\theta})cos(\psi) + m_p a(\dot{\psi}+\dot{\theta})^2 sin(\psi)]\}\\&+(m_s+m_p)(m_p a^2 + I_p)(\ddot{\psi}+\ddot{\theta}) + (m_s+m_p)\kappa\dot{\psi} = 0.\end{aligned} \tag{4.35}$$

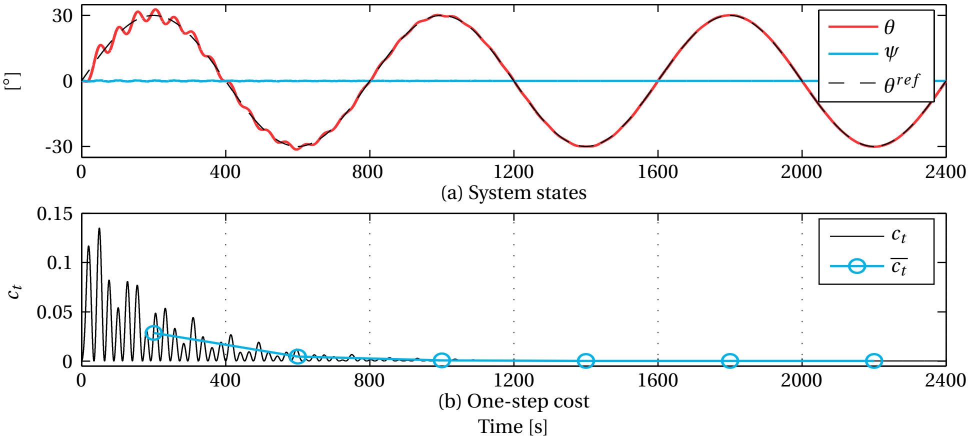
Thus, the MIMO nonlinear model of the satellite attitude control problem has been generated for simulations. The parameters used in the online learning simulations are $m_s = 600kg$, $I_s = 720kg/m^2$, $m_p = 100kg$, $I_p = 90kg/m^2$, $a = 0.3m$, $b = 0.3m$, $F_s = 500N$, and $\kappa = 0.19kg \cdot m^2/s$.

Results and Discussions

The target of this attitude control problem is to track a sinusoidal reference of θ while stabilizing ψ, which represent the liquid sloshing. The system model and even the thrust F_s is unknown, but the measurements of the system state $x = [\theta, \psi, \dot{\theta}, \dot{\psi}]^T$ and the control input $\mathbf{u} = [f_s, M_s]^T$ are available. The force and moment directly act on the rigid satellite and will slosh the liquid around. The liquid will start interacting with the satellite, which increases the system complexity and control difficulties. It is almost impossible to identify the global system model online and generate a proper actor before the system state diverges in this simulation.

Therefore, this section only applies the IHDP method to this satellite attitude control problem online without any a priori knowledge of the system. Figure 4.11 presents the online control performance in tracking the dynamical reference with zero initial tracking error $\Delta\theta_0 = 0$. The sub-figure on the top shows the system states θ and ψ during online tracking the sinusoidal signal with the amplitude of 30°. This simulation runs 3 periods of the sinusoidal signal, which is 2400 seconds. The sub-figure on the bottom provides not only the one-step cost at each time step c_t but also the averaged one-step cost over each 400 seconds $(\pi/2)$ $\overline{c_t}$.

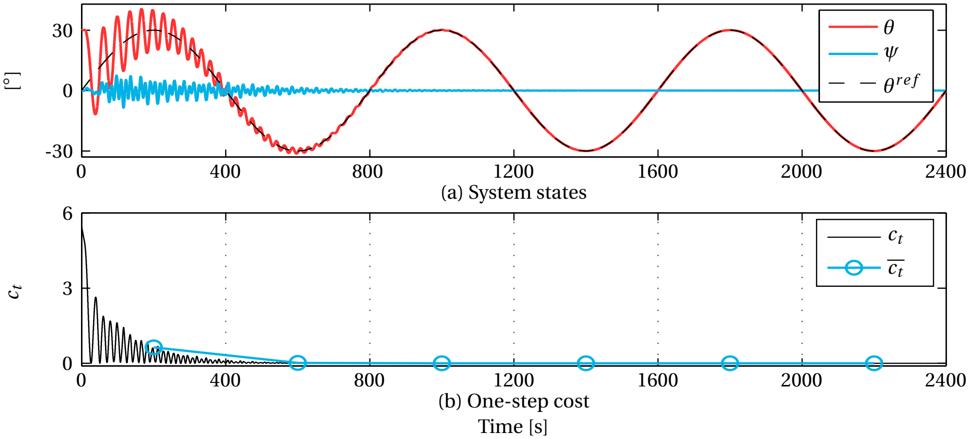
Initial tracking error in satellite angle $\Delta\theta_0$ may cause extra oscillations, system uncertainties, and even divergence if the online adaptability of the controller is not efficient and fast enough. Figures 4.12 and 4.13 present the online performance of the IHDP method with initial tracking errors ±30°, respectively. As seen in these figures, the IHDP method can deal with different initial tracking error without loss of accuracy.



(a) System states

(b) One-step cost

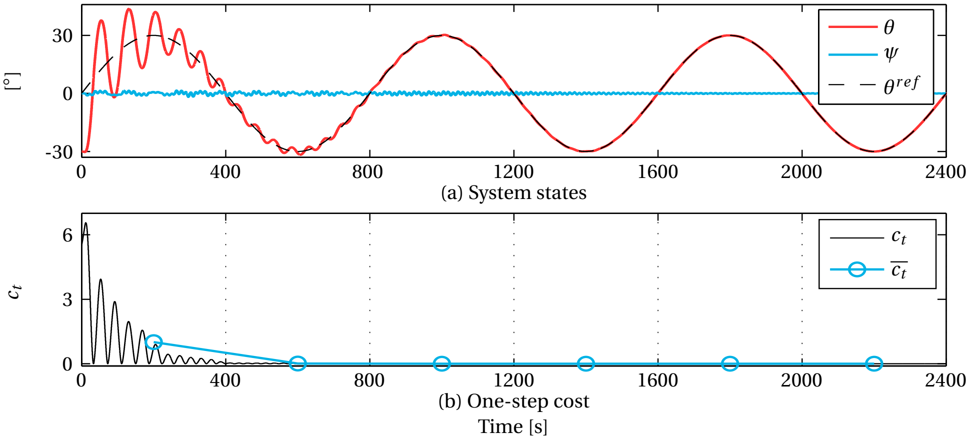
Figure 4.11: Online tracking control with zero initial states using the IHDP approach.



(a) System states

(b) One-step cost

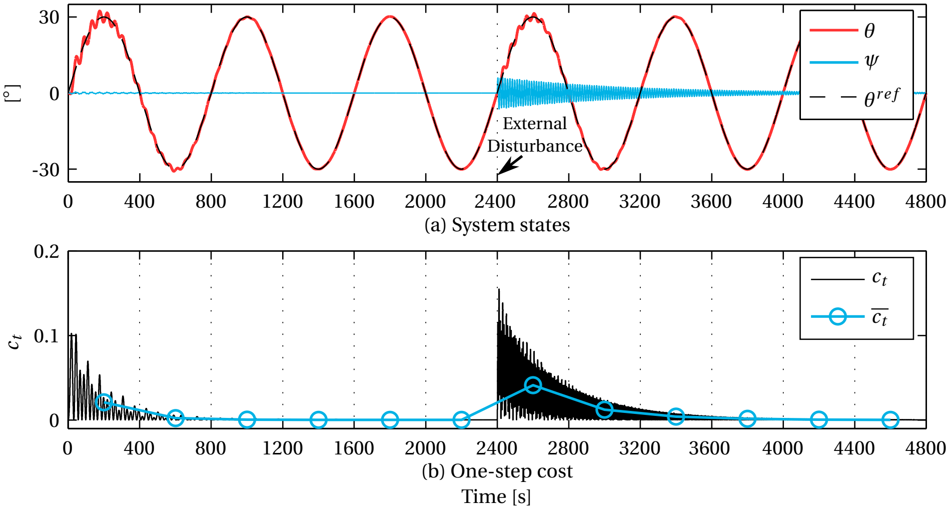
Figure 4.12: Online tracking control with a positive initial satellite angle $\theta_0 = 30^\circ$ using the IHDP approach.



(a) System states

(b) One-step cost

Figure 4.13: Online tracking control with a positive initial satellite angle $\theta_0 = -30^\circ$ using the IHDP approach.

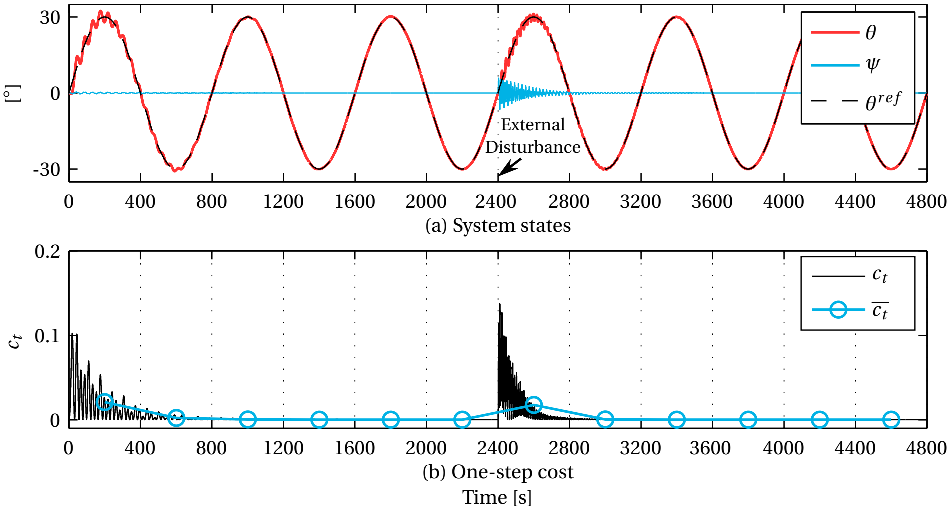


(a) System states

(b) One-step cost

Time [s]

Figure 4.14: Disturbance response without further adaptation when the disturbance occurs.



(a) System states

(b) One-step cost

Time [s]

Figure 4.15: Disturbance response with consistent adaptation.

To further validate the adaptability and robustness of the proposed IHDP method, another simulation is carried out with sudden external disturbance. One of the common disturbances on satellite is the external force. In this simulation, the external force f_s^{ext} acts through the satellite center of mass as same as f_s. It is $-100N$, placed at 2400 seconds and lasts for 1 second. This force brings an immediate oscillation to the liquid sloshing angle ψ, which also expands the experienced state space. Therefore, the control policy needs to adapt itself to stabilize this oscillation. Figure 4.14 presents the disturbance response without any adaptation of the actor by freezing the actor weights after 2400 seconds. The oscillation is damped out very slowly. This result also indicates that even the off-line learning can generate an exceptional policy in the experienced situations, the changes and uncertainties in the system or in the environment may degrade the control performance without online adaptation.

The sudden external disturbances may excite the local model identification immediately, adapt the cost function in the just experienced state space, and consequently self-adjust the control policy by updating the actor. As shown in Fig. 4.15, the IHDP method with further adaptation rejects the disturbance and improves the closed-loop performance much quicker compared to the result in Fig. 4.14. This result validates that the IHDP method is robust not only to internal uncertainties but also to external disturbances.

4.5. CONCLUSION

This chapter proposed a new approach, Incremental model based Heuristic Dynamic Programming (IHDP), to design adaptive controllers without a priori knowledge of the system dynamics. This method adopts the basic structure and adaptation rules of the Heuristic Dynamic Programming (HDP), which has a general nonlinear cost function. On the other hand, this method replaces the global system model approximator with an incremental model reconstructed by using a series of the most recent measurements. This linear model provides more accurate local system dynamics and state prediction and simplifies the actor update process. Therefore, this method does not need off-line training stage and may accelerate online learning. To accelerate the online learning is of great practical value, especially when a priori knowledge is unknown, the system is initially unstable, or the system or environment changes suddenly.

To compare the online control performance of the IHDP method and the conventional HDP method, this chapter applied both methods to a tracking problem on an unknown missile model with nonlinear aerodynamic uncertainties. The results show that the presented IHDP method speeds up the online learning, has a higher precision, and can deal with a wider range of initial states than the traditional HDP method. In addition, this chapter also applied the IHDP method to a more complex multiple-in multiple-out satellite attitude tracking control disturbed by liquid sloshing. The simulation results also demonstrate that the IHDP method is adaptive and robust to internal uncertainties and external disturbances.

As an extension of the incremental Approximate Dynamic Programming (iADP) method that uses a quadratic cost function, the IHDP method presented in this chapter separates the policy evaluation and improvement into two nonlinear structures, which are artificial neural networks in this chapter. Although these two approximations update alternatively based on each other, their adaptation is not necessarily synchronized. In another word, the critic update usually happens earlier than the actor. This time difference also provides a chance for the critic to evaluate a relatively consistent policy and makes it possible to prevent the initial off-line learning stage. This study generalized the use and applications of the iADP methods. Further research should, therefore, concentrate on the investigation of different types of approximators, experimentation of higher degree-of-freedom and more realistic applications, and further extensions to partially observable conditions.

5

Incremental Model Based Dual Heuristic Programming

The previous chapter presented the Incremental model based Heuristic Dynamic Programming (IHDP) method, which exploits incremental models instead of a global system model, as in conventional HDP methods. This IHDP method shows the advantages of using incremental models in Adaptive Critic Designs (ACDs), which make them promising solutions for control of aerospace systems. To further accelerate the convergence and improve the control performance, this chapter develops another online ACD method: Incremental model based Dual Heuristic Programming (IDHP). This chapter begins with a brief introduction of the conventional DHP algorithm and then focuses on the development of the IDHP method, in Section 5.2. Then, Section 5.3 describes the nonlinear missile model and discusses some implementation-related issues. Section 5.4 validates the IDHP method in an online reference tracking task and an online Fault-Tolerant Control (FTC) task, even with high-frequency measurement noise. The results demonstrate that the IDHP method can further increase the convergence rate and control performance compared to the DHP method, and can successfully control a faulty and unstable system in real-time.

This chapter is based on the following article:
Y. Zhou, E. van Kampen, and Q. P. Chu. Incremental model based online dual heuristic programming for nonlinear adaptive control. Control Engineering Practice. Vol. 73, p. 13-25, 2018. https://doi.org/10.1016/j.conengprac.2017.12.011 [106] .

Dual heuristic programming has gained an increasing interest in recent years because it provides an effective process for optimal adaptive control of uncertain nonlinear systems. However, it requires a global system model, which usually needs to be trained off-line. This chapter presents a new and efficient approach for online self-learning control based on dual heuristic programming. This method uses a recursive least square method to online identify an incremental model of the system instead of a global system model. The presented incremental model based dual heuristic programming method can adaptively generate a near-optimal controller online without a priori information of the system dynamics or an off-line training stage. To compare the online adaptability of the conventional dual heuristic programming method and the newly proposed method, two numerical experiments are performed: an online reference tracking task and a fault-tolerant control task. The results reveal that the proposed method outperforms the conventional dual heuristic programming method in online learning capacity, efficiency, accuracy, and robustness.

5.1. Introduction

Adaptive control strategies are the foundation for controlling nonlinear systems with uncertainties. To solve these control problems, Reinforcement Learning (RL) offers an option without using accurate system models [34]. RL is a self-learning method, in which actions are trained in order to minimize the cost-to-go from interaction with the environment. These self-learning methods link bio-inspired artificial intelligence techniques to the field of optimal control and adaptive control to overcome some of the limitations and challenges of traditional model-based control methods. Approximate Dynamic Programming (ADP) is an RL method aiming to solve optimal control problems with large or continuous state spaces [32, 50, 53, 101–103]. They apply an approximation of the true cost-to-go of states and/or an approximation towards the optimal control policy so as to tackle the 'curse of dimensionality'. Therefore, these methods belong to optimal adaptive control [34], and the trained policy is near-optimal.

As a class of ADP methods, Adaptive Critic Designs (ACDs) have shown great success in optimal adaptive control of nonlinear problems [32, 34, 50, 51, 107]. They are also known as Actor-Critics (ACs) because they separate evaluation (critic) and improvement (actor) using parametric structures. The critic adopts Temporal Difference (TD) methods to update the cost function, while the actor adapts its parameters towards the optimal policy by applying the principle of optimality [31, 34]. Although they are called ACs, they often need an extra structure to approximate the global system model so as to close the update path of the actor, the critic, or both. The critic, actor, and system model can be implemented with nonlinear function approximators, such as Artificial Neural Networks (ANN). With these approximators, ACDs can identify the system dynamics globally and then adaptively generate the control laws.

ACDs can generally be categorized into three groups: 1) Heuristic Dynamic Programming (HDP), which is the most basic form and uses the critic to approximate the true cost-to-go; 2) Dual Heuristic Programming (DHP), in which the critic approximates the derivatives of the true cost-to-go with respect to the critic inputs; and 3) Globalized Dual Heuristic Programming (GDHP), which approximates both the true cost-to-go and its derivatives. Several studies comparing the before-mentioned ACDs have shown that both DHP and GDHP outperform HDP in success rate and precision [51, 52]. The main reason is that the critic of the DHP and the GDHP directly outputs the derivatives of the cost function, which reduces the error introduced by the derivation backward through the critic of the HDP [55]. Although the GDHP did not show distinct advantages over the DHP, the computational complexity is considerably higher due to the second derivative terms [51, 55]. Therefore, the proposed method in this chapter is mainly related to the DHP.

In addition, Action Dependent (AD) variations of these three original versions have been developed by directly connecting the output of the actor to the input of the critic [51, 53, 54, 56]. The AD forms may reduce the dependency on the system model. From the theoretical point of view, the actor output is not necessarily an input to the critic, which estimates the value/cost function; from the practical perspective, the extra input can increase the complexity of the critic. Furthermore, previous studies comparing HDP and its AD form have reported that HDP controllers have a higher success rate in an auto-landing task [51], besides which it can operate in a wider range of flight conditions and adapts faster to the changed plant dynamics in controlling an F-16 aircraft model [54].

Online learning control with ACDs has been studied for years and is still one of the most active areas in RL today. However, ACDs often have two learning phases [50, 51, 53, 54, 108]: off-line learning and online learning. The main reason is that the identification of the global system model is not a trivial task, which needs certain time and usually an off-line learning phase beforehand [8, 9, 23–27]. During the online phase, extra computing cost is required to adaptively perform the approximation of the system with unforeseen dynamics, such as the resulting changes from the changes in the actor, a time-varying component in the system, uncertainties in the environment, and unexpected changes due to failures. Several studies [55, 56] have suggested to remove the global system model and to exploit previous critic outputs and/or inputs instead. Although this technique has been successfully applied to many ACD methods, it can only relieve the off-line learning phase of the AD forms. An accurate global system model still plays an important role in most ACDs, especially in DHP and GDHP because the update of both the critic and the actor depends on the system model.

This chapter proposes a systematic approach to developing online ACD controllers, more specifically for DHP, based on the incremental control technique. This incremental technique has been successfully applied to design adaptive controllers, such as Incremental Nonlinear Dynamic Inversion (INDI) [63, 65], Incremental BackStepping (IBS) [67] and incremental adaptive sliding mode control [109], to deal with system nonlinearities. However, these methods have not addressed optimization or synthesis of designed closed-loop systems. Incremental Approximate Dynamic Programming (iADP) [46, 80] was proposed for off-line near-optimal control of unknown nonlinear systems without using system models. This approach uses a quadratic function to approximate the cost function. Therefore, it is suitable only for systems with approximately convex cost functions.

In this chapter, Incremental model based Dual Heuristic Programming (IDHP) is developed for online adaptive control of unknown nonlinear systems. It uses a linear time-varying approximation of the original system to replace the global system model in conventional DHP. In addition, a Recursive Least Square (RLS) technique is used to identify the incremental model when assuming a sufficiently high sample rate for discretization. This method belongs to model-free control because it does not need any a priori information of the system dynamics at the beginning nor online identification of the global nonlinear system, but only the online identified incremental model.

The remainder of this chapter is structured as follows. Section 5.2 starts with a brief introduction of the conventional DHP algorithm and then focuses on the development of the IDHP method. Section 5.3 introduces the nonlinear air vehicle model and discusses some related issues to achieve the implementation of the DHP and IDHP methods. Then, section 5.4 applies these two algorithms to two illustrative control tasks and compares their performance with regard to success rates, tracking errors, settling time, and robustness in different initial states and failures. Lastly, section 5.5 concludes the advantages and disadvantages of using the incremental approach with DHP and addresses the challenges and possibilities of the future research.

5.2. INCREMENTAL MODEL BASED DUAL HEURISTIC PROGRAMMING DESIGN

This section develops an online adaptive controller for unknown nonlinear systems, namely Incremental model based Dual Heuristic Programming (IDHP). The major difference with the conventional DHP is that IDHP does not use a nonlinear function approximator to approach the global system model. Instead, it exploits an online identified incremental model. Therefore, this method can adapt the controller online without a priori knowledge of the system dynamics or off-line learning of the system model. The rest of this section will briefly introduce the conventional DHP and then focus on the IDHP algorithm and adaptation rules.

5.2.1. DHP FRAMEWORK AND GLOBAL SYSTEM MODEL

DHP methods are most favored within the ACD category because they have higher success rate and accuracy than HDP and lower computational complexity than GDHP [51, 52]. Conventional DHP controllers use three nonlinear function approximators to approach the actor, the critic, and the system dynamical model with weights (or more generally called model parameters) $\mathbf{w}_a$, $\mathbf{w}_c$, and $\mathbf{w}_m$, respectively, as shown in Fig.5.1. The Back-Propagation (BP) algorithms of both the critic and the actor are based on the system model.

The DHP method uses the system model to approximate the dynamics of the global system. The inputs of the system model are the current state, $\mathbf{x}_t \in \mathscr{R}^n$, and the control input, $\mathbf{u}_t \in \mathscr{R}^m$, based on which it outputs the estimated next state, $\hat{\mathbf{x}}_{t+1} \in \mathscr{R}^n$. The system model weights $\mathbf{w}_m(t)$ are updated by minimizing the model error. The error is defined as the difference between the measured state $\mathbf{x}_t$ and the estimated state $\hat{\mathbf{x}}_t$:

$$E_m(t) = \frac{1}{2}\mathbf{e}_m(t)^T\mathbf{e}_m(t), \tag{5.1}$$

where

$$\mathbf{e}_m(t) = \mathbf{x}_t - \hat{\mathbf{x}}_t. \tag{5.2}$$

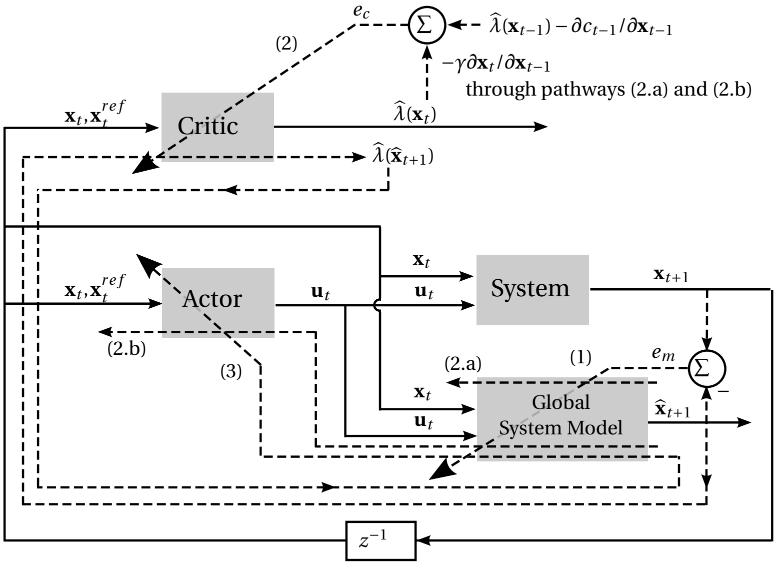

e_c
(2)
Σ
$\hat{\lambda}(\mathbf{x}_{t-1}) - \partial c_{t-1}/\partial \mathbf{x}_{t-1}$
$-\gamma\partial\mathbf{x}_t/\partial\mathbf{x}_{t-1}$
through pathways (2.a) and (2.b)
$\mathbf{x}_t, \mathbf{x}_t^{ref}$
Critic
$\hat{\lambda}(\mathbf{x}_t)$
$\hat{\lambda}(\hat{\mathbf{x}}_{t+1})$
$\mathbf{x}_t$
$\mathbf{u}_t$
System
$\mathbf{x}_{t+1}$
$\mathbf{x}_t, \mathbf{x}_t^{ref}$
Actor
$\mathbf{u}_t$
(2.b)
(3)
(2.a)
(1)
e_m
Σ
$-$
$\mathbf{x}_t$
$\mathbf{u}_t$
Global System Model
$\hat{\mathbf{x}}_{t+1}$
z^{-1}

Figure 5.1: A schematic diagram of DHP using a global system model, where the solid lines represent the feedforward flow of signals, and the dashed lines represent the BP pathways.

The update rule for system model weights are formulated according to the gradient-descent algorithm with a learning rate η_m:

$$\mathbf{w}_m(t+1) = \mathbf{w}_m(t) + \Delta\mathbf{w}_m(t), \tag{5.3}$$

$$\Delta\mathbf{w}_m(t) = -\eta_m \cdot \frac{\partial E_m(t)}{\partial \hat{\mathbf{x}}_t} \frac{\partial \hat{\mathbf{x}}_t}{\partial \mathbf{w}_m(t)}. \tag{5.4}$$

Nonlinear function approximators, such as artificial neural networks, can identify the system dynamics globally. However, online identification of the global model is not a trivial task. It needs a certain time and effort, depending on the complexity and nonlinearity of the system, to approximate a feasible model. Therefore, conventional DHP usually needs an off-line stage beforehand to identify the global system model. This stage makes the DHP an off-line method and less robust in the presence of sudden changes of the system dynamics. On the other hand, incremental model techniques provide a quick approximation of the locally linearized model. It will not only make the algorithm online but also reduce the computational burden of the whole adaptation process.

5.2.2. IDHP FRAMEWORK AND ADAPTATION RULES

The IDHP method is devised based on the DHP method and incremental control techniques. As illustrated in Fig. 5.2, the IDHP controller only has two nonlinear function approximators to approach the actor and the critic. The system information to update the critic and the actor can be obtained from the incremental model identified immediately after a few measurements. Therefore, the design of the IDHP controller relieves the off-line learning phase. This section will present and elaborate the IDHP method.

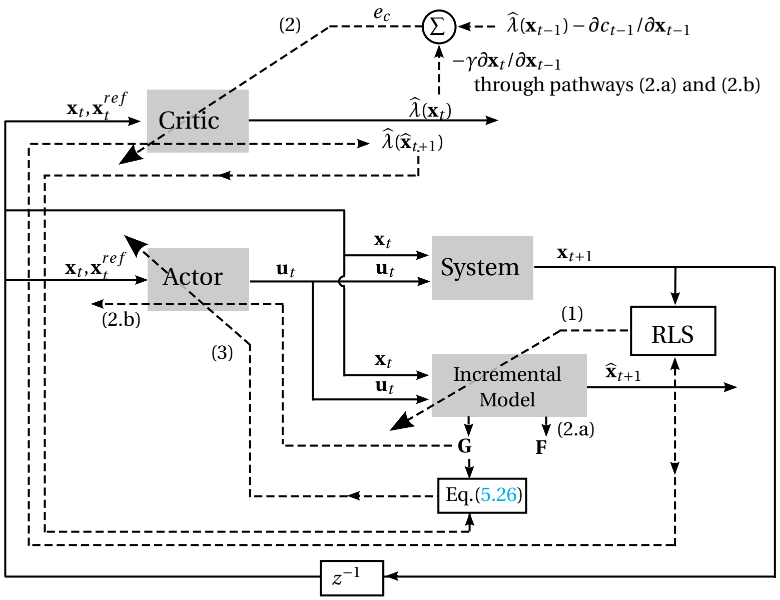

(2)
e_c
Σ
$\hat{\lambda}(\mathbf{x}_{t-1}) - \partial c_{t-1}/\partial \mathbf{x}_{t-1}$
$-\gamma \partial \mathbf{x}_t/\partial \mathbf{x}_{t-1}$
through pathways (2.a) and (2.b)
$\mathbf{x}_t, \mathbf{x}_t^{ref}$
Critic
$\hat{\lambda}(\mathbf{x}_t)$
$\hat{\lambda}(\hat{\mathbf{x}}_{t+1})$
$\mathbf{x}_t, \mathbf{x}_t^{ref}$
Actor
$\mathbf{u}_t$
$\mathbf{x}_t$
$\mathbf{u}_t$
System
$\mathbf{x}_{t+1}$
(2.b)
(3)
(1)
RLS
$\mathbf{x}_t$
$\mathbf{u}_t$
Incremental Model
$\hat{\mathbf{x}}_{t+1}$
G
F
(2.a)
Eq.(5.26)
z^{-1}

Figure 5.2: A schematic diagram of the IDHP using a time-varying incremental model, where the solid lines represent the feedforward flow of signals, and the dashed lines represent the adaptation pathways.

Incremental Model

The incremental model at the current time can be approximated by using the conditions of the system in an instant before [46, 65]. This technique has been successfully applied to deal with system nonlinearity assuming sufficiently high sample rate [63, 65, 67]. The dynamic and kinematic equations of a nonlinear system can generally be given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t)], \tag{5.5}$$

where $f[\mathbf{x}(t), \mathbf{u}(t)] \in \mathscr{R}^n$ provides the physical evaluation of the state vector over time.

The system dynamics at time t approaching t_0 can be linearized approximately by using the first-order Taylor series expansion:

$$\dot{\mathbf{x}}(t) \approx \dot{\mathbf{x}}(t_0) + \mathbf{F}[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{x}(t) - \mathbf{x}(t_0)] + \mathbf{G}[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{u}(t) - \mathbf{u}(t_0)], \tag{5.6}$$

where $\mathbf{F}[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{x}(t)}|_{\mathbf{x}(t_0), \mathbf{u}(t_0)} \in \mathscr{R}^{n \times n}$ is the system matrix of the linearized model, and $\mathbf{G}[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{u}(t)}|_{\mathbf{x}(t_0), \mathbf{u}(t_0)} \in \mathscr{R}^{n \times m}$ is the control effectiveness matrix. When the states and state derivatives of the system are assumed to be measurable, i.e., $\Delta\dot{\mathbf{x}}(t), \Delta\mathbf{x}(t), \Delta\mathbf{u}(t)$ are measurable, the system model around time t_0 can be written in an incremental form:

$$\Delta\dot{\mathbf{x}}(t) \approx \mathbf{F}[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{x}(t) + \mathbf{G}[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{u}(t), \tag{5.7}$$

where $\Delta\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(t) - \dot{\mathbf{x}}(t_0)$, $\Delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}(t_0)$, and $\Delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}(t_0)$.

Although most physical systems are continuous, their measurements are always discrete. With a constant, high data sampling frequency, nonlinear systems can be written in a discrete form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{5.8}$$

where $f(\mathbf{x}_t, \mathbf{u}_t) \in \mathscr{R}^n$ provides the system dynamics. When the sample time Δt is sufficiently small, the system dynamics around $\mathbf{x}_t$ can also be linearized by taking the Taylor expansion:

$$\mathbf{x}_{t+1} \approx \mathbf{x}_t + \mathbf{F}_{t-1} \cdot (\mathbf{x}_t - \mathbf{x}_{t-1}) + \mathbf{G}_{t-1} \cdot (\mathbf{u}_t - \mathbf{u}_{t-1}), \tag{5.9}$$

where $\mathbf{F}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times n}$ is the system transition matrix, and $\mathbf{G}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathscr{R}^{n\times m}$ is the input distribution matrix at time step $t-1$ for discretized systems. The incremental form of this discrete nonlinear system can be obtained as follows:

$$\Delta\mathbf{x}_{t+1} \approx \mathbf{F}_{t-1}\Delta\mathbf{x}_t + \mathbf{G}_{t-1}\Delta\mathbf{u}_t, \tag{5.10}$$

With the high-frequency sample data and the relatively slow-varying system assumption, the current linearized model is time-varying. This model needs to be available online to obtain the approximated value of $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}|_m$ and $\frac{\partial \mathbf{x}_t}{\partial \mathbf{u}_{t-1}}|_m$ without using a global nonlinear system model. Therefore, an online identification of the time-varying matrices $\widehat{\mathbf{F}}_{t-1}$ and $\widehat{\mathbf{G}}_{t-1}$ is required.

THE CRITIC

Although the function and structure of the critic in the IDHP method is similar as in the DHP method, the calculation burden is reduced by directly applying the incremental model. The critic approximates the derivatives of the true cost-to-go function $J(\mathbf{x}_t)$ with respect to the state vector $\mathbf{x}_t$:

$$\lambda(\mathbf{x}_t) = \frac{\partial J(\mathbf{x}_t)}{\partial \mathbf{x}_t}. \tag{5.11}$$

The true cost-to-go of the current state $\mathbf{x}_t$ is the cumulative future cost:

$$J(\mathbf{x}_t) = \sum_{l=t}^{\infty} \gamma^{l-t} c_l, \tag{5.12}$$

where $\gamma \in [0,1]$ is a scalar called *discount factor* or *forgetting factor*, and c_l is the one-step cost at a future time l. The cost c_l is often a function of the state and/or the control input at that time, as follows:

$$c_l = c(\mathbf{x}_l, \mathbf{x}_l^{ref}) = (\mathbf{x}_l - \mathbf{x}_l^{ref})^T \mathbf{Q}(\mathbf{x}_l - \mathbf{x}_l^{ref}), \tag{5.13}$$

where $\mathbf{Q} \in \mathscr{R}^{n\times n}$ is a positive definite matrix. To normalize the effect of each state, the normalization factors are used in the $\mathbf{Q}$ matrix: $\mathbf{Q} = diag\{\zeta_1, \zeta_2, ..., \zeta_n\}$, where ζ is a weight indicating the importance of the cost for the related state.

The error function for the critic is defined according to the TD error as shown below:

$$E_c(t) = \frac{1}{2}\mathbf{e}_c(t)^T \mathbf{e}_c(t), \tag{5.14}$$

where

$$\begin{aligned}\mathbf{e}_c(t) &= \frac{\partial[J(\mathbf{x}_{t-1}) - c_{t-1} - \gamma J(\mathbf{x}_t)]}{\partial \mathbf{x}_{t-1}} \\ &= \lambda(\mathbf{x}_{t-1}) - \frac{\partial c_{t-1}}{\partial \mathbf{x}_{t-1}} - \gamma\lambda(\mathbf{x}_t)\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}.\end{aligned} \tag{5.15}$$

In Eq. (5.15), $\lambda(\mathbf{x}_{t-1})$ and $\lambda(\mathbf{x}_t)$ can be approximated by $\widehat{\lambda}(\mathbf{x}_{t-1})$ and $\widehat{\lambda}(\mathbf{x}_t)$, which are calculated through the critic with weights at the current time $\mathbf{w}_c(t)$. The system state is a function of the previous state and the control input. This can be approximated by a global system model in conventional DHP method or by an incremental model in the proposed IDHP method. Therefore, the last term in Eq. (5.15), $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}$, need to be calculated through two pathways, pathways (2.a) and (2.b) as shown in both Figs. 5.1 and 5.2:

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}} = \underbrace{\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}|_m}_{pathway(2.a)} + \underbrace{\frac{\partial \mathbf{x}_t}{\partial \mathbf{u}_{t-1}}|_m \cdot \frac{\partial \mathbf{u}_{t-1}}{\partial \mathbf{x}_{t-1}}|_a}_{pathway(2.b)}. \tag{5.16}$$

IDHP method directly applies the incremental model information to approximate the two system model derivative terms in Eq. (5.16), which are all supposed to be calculated back through the global system model in conventional DHP methods. Therefore, IDHP can simplify the calculation of Eq. (5.16) as follows:

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}} \approx \widehat{\mathbf{F}}_{t-1} + \widehat{\mathbf{G}}_{t-1} \cdot \frac{\partial \mathbf{u}_{t-1}}{\partial \mathbf{x}_{t-1}}|_a. \tag{5.17}$$

The critic weights can be updated with a learning rate η_c to minimize the error $E_c(t)$:

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \Delta\mathbf{w}_c(t), \tag{5.18}$$

$$\begin{aligned}\Delta\mathbf{w}_c(t) &= -\eta_c \cdot \frac{\partial E_c(t)}{\partial \widehat{\lambda}(\mathbf{x}_{t-1})} \cdot \frac{\partial \widehat{\lambda}(\mathbf{x}_{t-1})}{\partial \mathbf{w}_c(t)} \\ &= -\eta_c \cdot \mathbf{e}_c(t)^T \cdot \frac{\partial \widehat{\lambda}(\mathbf{x}_{t-1})}{\partial \mathbf{w}_c(t)}.\end{aligned} \tag{5.19}$$

THE ACTOR

Similar to the critic, the actor adaptation in the IDHP is also simplified by involving the incremental model instead of the global system model. The control policy is improved through updating the actor to minimizes the non-negative cost-to-go, $J(\mathbf{x}_t)$:

$$\begin{aligned}\mathbf{u}_t^* &= \arg\min_{\mathbf{u}_t} J(\mathbf{x}_t) \\ &= \arg\min_{\mathbf{u}_t} [c_t + \gamma J(\mathbf{x}_{t+1})].\end{aligned} \tag{5.20}$$

By applying the gradient descent method, a weight update expression can be written as follows:

$$\mathbf{w}_a(t+1) = \mathbf{w}_a(t) + \Delta\mathbf{w}_a(t), \tag{5.21}$$

$$\begin{aligned}\Delta \mathbf{w}_a(t) &= -\eta_a \cdot \frac{\partial J(\mathbf{x}_t)}{\partial \mathbf{u}_t}\frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)} \\ &= -\eta_a \cdot \left[\frac{\partial c_t}{\partial \mathbf{u}_t} + \gamma\lambda(\mathbf{x}_{t+1})\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}\right]\frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}.\end{aligned} \tag{5.22}$$

The weight update of the actor involves the critic and the system model, through the 3rd back-propagation direction, as shown in Figs. 5.1 and 5.2. The incremental model can output the estimation of the next state $\widehat{\mathbf{x}}_{t+1}$ for an input $\mathbf{u}_t$. This also helps to get the useful term $\frac{\partial \widehat{\mathbf{x}}(t+1)}{\partial \mathbf{u}_t}$, which approximates $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$, in updating the actor [54]. Therefore, Eq. (5.22) can be rewritten as follows:

$$\Delta \mathbf{w}_a(t) = -\eta_a \cdot \left[\frac{\partial c_t}{\partial \mathbf{u}_t} + \gamma\widehat{\lambda}(\widehat{\mathbf{x}}_{t+1})\frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t}|_m\right]\frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}. \tag{5.23}$$

In the forward calculation, the next state can be predicted using the identified incremental model as follows:

$$\widehat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \widehat{\mathbf{F}}_{t-1}\Delta\mathbf{x}_t + \widehat{\mathbf{G}}_{t-1}\Delta\mathbf{u}_t. \tag{5.24}$$

In the back-propagation calculation, the derivative of the next state with respect to the control input, $\frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t}$, can be approximated using the incremental model information:

$$\frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t} \approx \widehat{\mathbf{G}}_{t-1}. \tag{5.25}$$

Therefore, Eq. (5.23) can be further simplified as follows:

$$\Delta \mathbf{w}_a(t) = -\eta_a \cdot \left[\frac{\partial c_t}{\partial \mathbf{u}_t} + \gamma\widehat{\lambda}(\widehat{\mathbf{x}}_{t+1})\widehat{\mathbf{G}}_{t-1}\right]\frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}. \tag{5.26}$$

5.2.3. INCREMENTAL MODEL IDENTIFICATION

Assuming that the incremental model is identifiable using LS techniques with measurements of proper excitation and response, this chapter adopts the Recursive Least Square (RLS) approach to online identify the system transition matrix $\mathbf{F}_{t-1}$ and the input distribution matrix $\mathbf{G}_{t-1}$ of the linearized model. The incremental form of the state in Eq. (5.10) can be rewritten row by row as follows:

$$\Delta x_{r,t+1} \approx \begin{bmatrix}\Delta\mathbf{x}_t^T & \Delta\mathbf{u}_t^T\end{bmatrix}\cdot\begin{bmatrix}\mathbf{f}_r^T \\ \mathbf{g}_r^T\end{bmatrix}, \tag{5.27}$$

where $\Delta x_{r,t+1} = x_{r,t+1} - x_{r,t}$ is the increment of rth state element, $\mathbf{f}_r$ and $\mathbf{g}_r$ are the elements of rth row vector of $\mathbf{F}_{t-1}$ and $\mathbf{G}_{t-1}$. These parameters can be identified using the RLS usually row by row. Since they share the same covariance matrix, they can also be identified together as in the parameter matrix $\Theta_{t-1} = \begin{bmatrix}\mathbf{F}_{t-1}^T \\ \mathbf{G}_{t-1}^T\end{bmatrix} \in \mathscr{R}^{(n+m)\times n}$.

The state prediction equation can be written as follows:

$$\Delta\widehat{\mathbf{x}}_{t+1}^T = X_t^T\widehat{\Theta}_{t-1}, \tag{5.28}$$

where $X_t = \begin{bmatrix} \Delta\mathbf{x}_t \\ \Delta\mathbf{u}_t \end{bmatrix} \in \mathscr{R}^{(n+m)\times 1}$ stands for the input information of the incremental model. The RLS approach adopted in this chapter is presented as follows [99]:

$$\epsilon_t = \Delta\mathbf{x}_{t+1}^T - \Delta\widehat{\mathbf{x}}_{t+1}^T, \tag{5.29}$$

$$\widehat{\Theta}_t = \widehat{\Theta}_{t-1} + \frac{Cov_{t-1}X_t}{\gamma_{RLS} + X_t^T Cov_{t-1} X_t}\epsilon_t, \tag{5.30}$$

$$Cov_t = \frac{1}{\gamma_{RLS}}\left(Cov_{t-1} - \frac{Cov_{t-1}X_t X_t^T Cov_{t-1}}{\gamma_{RLS} + X_t^T Cov_{t-1} X_t}\right), \tag{5.31}$$

where $\epsilon_t \in \mathscr{R}^{1\times n}$ is the prediction error, also called *innovation*, $Cov_t \in \mathscr{R}^{(n+m)\times(n+m)}$ is the estimation covariance matrix, and γ_{RLS} is the forgetting factor for this RLS approach.

The RLS approach used in this chapter possesses a significant advantage over the piecewise Ordinary Least Square (OLS) method: RLS has fewer issues with persistent excitation. The OLS method needs to do a matrix inversion at each update [46, 91]. If there is not enough excitation at that moment, the matrix might not be invertible, and the parameters cannot be identified. The RLS method, on the other hand, does not need to do matrix inversions because it uses the matrix inversion lemma to update the covariance matrix, which contains the information of that inverted matrix. Therefore, it can effectively identify parameters of the time-varying system and also keeps the parameters relatively stable when the excitation is not enough. A comparison of these two methods will be made in section 5.4.2. This chapter initializes the **F** matrix as an identity matrix and the **G** matrix as a zero matrix and chooses the forgetting factor γ_{RLS} to be 0.8.

5.3. Flight Control Simulation

The first part in this section briefly introduces the nonlinear air vehicle model. The second part discusses the implementation of the aforementioned algorithms and some related issues, including the excitation of the system, the structured actor, and the learning rate.

5

5.3.1. Air Vehicle Model

Air vehicle models can be highly nonlinear, and their dynamic and kinematic state equations can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t) + \mathbf{w}(t)], \tag{5.32}$$

where $\mathbf{w}(t)$ is the external disturbance and is set to be caused only by the input noise.

This chapter applies the discussed methods to a simplified missile model obtained from [68, 69] to track a reference signal. There are several reasons for using this missile model: 1) This model is simple while nonlinear, which is suitable for a validation of the newly-developed model-free adaptive IDHP method. 2) For more complex models, conventional DHP will need off-line learning stage [50–53], which may impede a fair comparison to the IDHP method. 3) The missile model can operate at a high and rapidly changing angle of attack, which leads to a high nonlinearity. 4) It is a second-order continuous model, which means it is relatively real and complete. Although the model is only valid within the given flight envelope, the model can be still used out of the envelope in the failure occasions.

The nonlinear model of a short period flight control problem consists of two states: angle of attack α and pitch rate q (i.e., the system state vector is $\mathbf{x} = [\alpha \quad q]^T$). Only the pitch is controlled using elevator deflection δ_e. The nonlinear model in the pitch axis is simulated around a steady wings-level flight condition:

$$\dot{\alpha} = q + \frac{\bar{q}S}{m_a V_T} C_z(\alpha, q, M_a, \delta_e), \tag{5.33}$$

$$\dot{q} = \frac{\bar{q}Sd_l}{I_{yy}} C_m(\alpha, q, M_a, \delta_e), \tag{5.34}$$

where $\bar{q}$ is dynamic pressure, S is reference area, m_a is mass, V_T is speed, d_l is reference length, I_{yy} is pitching moment of inertia, C_z is the force coefficient in body Z-direction, and C_m is the pitch moment coefficient. C_z and C_m are nonlinear functions of angle of attack α, pitch rate q, Mach number M_a and elevator deflection δ_e.

The aerodynamic parameters of this model are valid for $-10° < \alpha < 10°$ [68, 69]:

$$\begin{aligned}
C_z(\alpha, q, M_a, \delta_e) &= C_{z1}(\alpha, M_a) + B_z\delta_e, \\
C_m(\alpha, q, M_a, \delta_e) &= C_{m1}(\alpha, M_a) + B_m\delta_e, \\
B_z &= b_1 M_a + b_2, \\
B_m &= b_3 M_a + b_4, \\
C_{z1}(\alpha, M_a) &= \phi_{z1}(\alpha) + \phi_{z2} M_a, \\
C_{z2}(\alpha, M_a) &= \phi_{m1}(\alpha) + \phi_{m2} M_a, \\
\phi_{z1}(\alpha) &= h_1\alpha^3 + h_2\alpha|\alpha| + h_3\alpha, \\
\phi_{m1}(\alpha) &= h_4\alpha^3 + h_5\alpha|\alpha| + h_6\alpha, \\
\phi_{z2} &= h_7\alpha|\alpha| + h_8\alpha, \\
\phi_{m2} &= h_9\alpha|\alpha| + h_{10}\alpha,
\end{aligned} \tag{5.35}$$

where $b_1, ..., b_4$, $h_1, ..., h_{10}$ are identified constant coefficients in the flight envelop, and the Mach number M_a is set to be 2.2.

5.3.2. Implementation Related Issues

In this chapter, all the nonlinear function approximators in both DHP and IDHP algorithms are MultiLayer Perceptrons (MLPs) neural networks, which consist of fully connected, feedforward layers of nodes. Each neural network has one hidden layer, where each node is a neuron with a hyperbolic tangent activation function bounded with $(-1, 1)$. Because the output of the hidden layer is multiplied with an output weight, the neural network with bias terms can theoretically approximate any value. The detailed neural network calculation and derivation have been provided in [91] and will not be carried out in this chapter.

The actor and the critic neural networks in the IDHP method have the same setting as the DHP method. In this chapter, the number of hidden layer neurons in the actor and the critic is 6, and in the system model network is 10. The neural network weights are updated using the Least Mean Square (LMS) method. To prevent the sudden growth to infinity, the limit of the neural network weights is $[-30, 30]$.

Cascaded Actor Network

To take advantages of the physical properties of the air vehicle system, a structured cascaded actor network [53, 54] is used as shown in Fig. 5.3. This cascaded structure separates the inner loop and outer loop control, which provides specific relationships between the angular rate and the attitude. As long as the concerned full states and control input of this air vehicle are known, this structure can be easily implemented in the actor network.

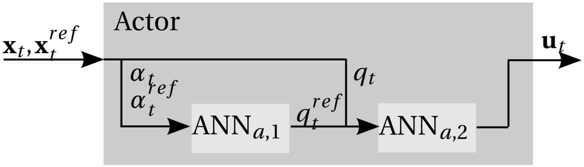


Figure 5.3: The architecture of the cascaded actor network.

Persistent Excitation

To accomplish the reference tracking task, an adaptive controller with the actor needs to be found out by minimizing the cost-to-go $J(\mathbf{x}_t)$ with a feasible critic and system model. As with other ADP methods, good evaluation depends heavily on the exploration of the state space, which is represented by Persistent Excitation (PE). Although RLS depends less on PE, PE is still imperative for identifying the incremental model. Many different input techniques can be used to excite aircraft modes, such as doublets, 3211 doublets, pseudo-random noise, and classical sine waves.

This chapter introduces an input disturbance, which is a sum of sinusoidal signals. This disturbance persistently excites the system for identification of the system and exploration of the state space in DHP methods. On the other hand, disturbances are usually undesirable inputs in the real world. Therefore, the flight control task is to track the reference signal as well as to stabilize the system, if there is any disturbance. Because of the online learning capability of DHP methods, the disturbance can be compensated without being identified.

Adaptive Learning Rate

DHP methods, similar to other ACDs, are online learning methods. They iteratively learn the actor and the critic, of which value depends on each other. Online learning in DHP uses the LMS techniques, which update the weights based only on the error at the current time. This method incrementally updates the neural network weights along the steepest descent with a learning rate η. The convergence to the optimal weights depends heavily on a properly chosen η. If η is chosen to be too small, the time to converge will be very large, and it may be trapped in a local optimum. If η is too large, the weight may change by a large amount and oscillate around the optimal weights.

To make the convergence less sensitive to the chosen learning rate, this chapter uses an adaptive learning rate η_t at time t. It is self-tuned in each update. A proper learning rate needs to meet the condition that the new weights decrease the network error. In each time-step, the initial learning rate is assigned by the previous time-step. This method calculates the direction of steepest descent $\frac{\partial E(t)}{\partial \mathbf{w}(t)}$ and searches along this line by halving the learning rate η_t until it meets the condition. With this η_t, the neural network weights can be updated as follows:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t \cdot \frac{\partial E(t)}{\partial \mathbf{w}(t)}. \tag{5.36}$$

This learning rate will also be assigned as the initial learning rate for the next step, if the network errors of the new weights and the old weights have different signs. Otherwise, the learning rate for the next time-step will be doubled. The initial learning rates $\eta_{0,0}$ at time $t = 0$ for the traditional DHP and the IDHP algorithms are set the same. They are chosen to be 10, 20, 10 for the system model, the critic, and the actor, respectively.

5.4. Results and Discussion

In this section, both the DHP and the IDHP algorithms are applied to a simulation of controlling a nonlinear air vehicle model for validation. First, the flight controller learns to track a changing reference in the presence of input disturbances, which is a most basic and essential control task for air vehicles. Then, these two methods are applied to failure cases, where the system suddenly changes and becomes unstable.

5.4.1. Online Reference Tracking

This section compares the conventional DHP and the IDHP methods by applying them to an online tracking problem. To be more specific, the controllers are required to control the angle of attack α to track the reference signal α^{ref}, which is a sine function of time with the amplitude of 10 degrees within 2 periods of the reference signal (4π seconds).

One of the objectives of these algorithms is to train the controller to track the reference regardless of the initial conditions of the air vehicle. Therefore, multiple numerical experiments are conducted with different initial conditions. Figures 5.4 to 5.6 compare the performance of DHP and IDHP with different initial states. Figure 5.4 shows their performance when the initial state is zero ($\alpha_0 = 0°$). In addition, Figs. 5.5 and 5.6 present the simulation results with positive ($\alpha_0 = 4°$) and negative ($\alpha_0 = -4°$) initial states. The subfigures (a) and (b) in Figs. 5.4 to 5.6 provide how these algorithms learn to track the reference signal online and their tracking errors during this task, respectively. As illustrated in these figures, IDHP method can identify the local model and reject the disturbance faster, and follow the reference signal more precisely.

Furthermore, IDHP method can deal with different initial states within the valid range of angle of attack ($\alpha \in [-10°, 10°]$) without loss of accuracy, which is visible in Fig. 5.7. On the other hand, the conventional DHP method cannot perform the online tracking task when the initial states are beyond the range $[-4°, 4°]$. This difference indicates that IDHP can deal with a wider range of the initial states compared to DHP methods.

Besides the initial states, initial weights of neural networks may also manipulate the learning. This chapter, therefore, examines different, random initial neural network

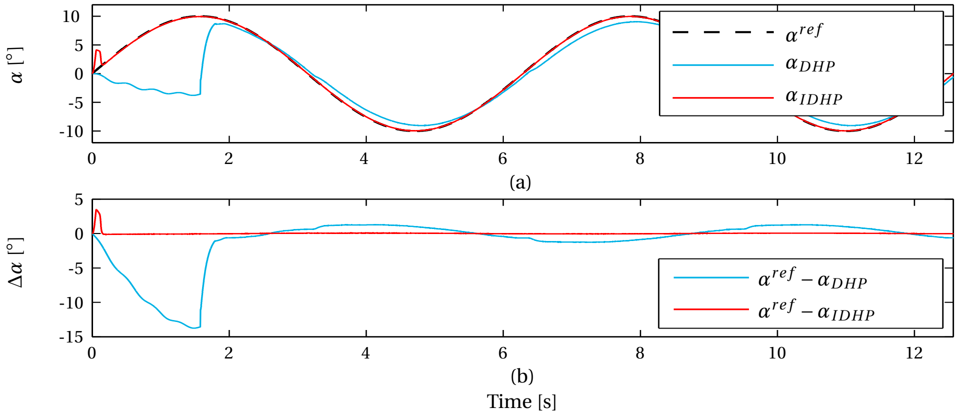


Figure 5.4: Online tracking control with the zero initial state using DHP and IDHP approaches.

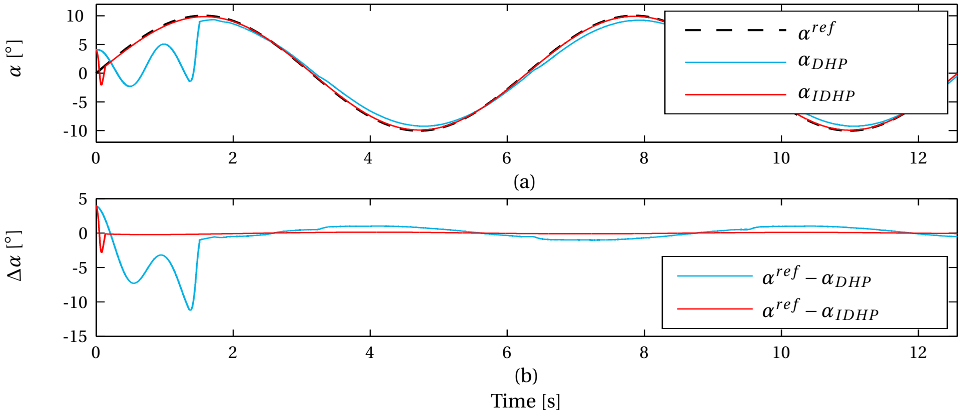


Figure 5.5: Online tracking control with a positive initial state, $\alpha_0 = 4°$, using DHP and IDHP approaches.

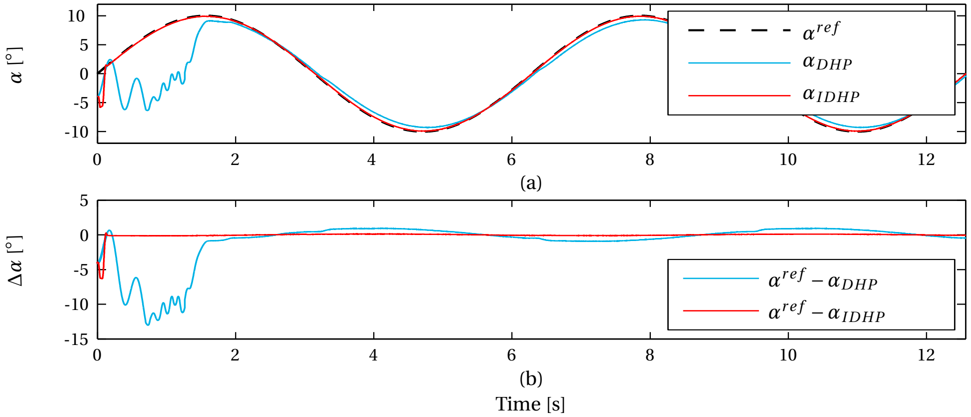


Figure 5.6: Online tracking control with a negative initial state, $\alpha_0 = -4°$, using DHP and IDHP approaches.

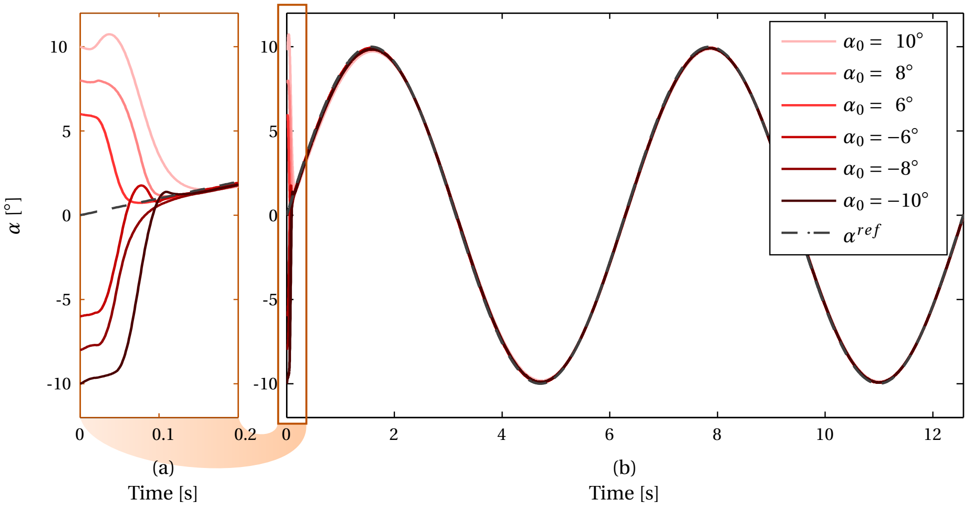


Figure 5.7: Online tracking control with different initial states using the IDHP approach.

weights from the range $(-0.01, 0.01)$. A successful trial is admitted if it converges within 1 period of the reference signal and the error between the target state and the reference signal is no more than 1.5 degrees hereafter. With these different initial weights as well as different initial states within $[-4°, 4°]$, the success rate is 95.5% for IDHP method and 80.9% for traditional DHP method. However, when the initial states are randomly chosen from the full valid range of angle of attack $[-10°, 10°]$, the success rate remains at the same level (93.3%) for IDHP method, while drops to 45.9% for traditional DHP method.

The tracking error and settling time for those successful trials are also analyzed as indicated in Table 5.1. The average tracking error is calculated within $(2\pi, 4\pi)$ seconds. In addition, the average settling time is defined as the time that the error reaches and remains within 1.5 degrees. It is apparent from this table that 1) the IDHP method spends averagely less time (0.103 seconds) to track the reference, compared to the conventional DHP (1.598 seconds), and 2) the average tracking error after 2π seconds is decreased from 0.722 degrees with conventional DHP to 0.059 degrees with IDHP method. This again validates that online IDHP method follows the reference signal faster and more precisely.

Table 5.1: Performance of DHP and IDHP in the online tracking task

	DHP	IDHP
Average Tracking Error	0.722°	0.059°
Average Settling Time	1.598 s	0.103 s

Consistent with findings in [51, 52] , the DHP method has less average settling time and higher accuracy than the HDP method [91]. The IDHP method, as expected, also outperforms the IHDP method [91] in both success rate, precision, and the range of the initial states when applied to the same online tracking task. In addition, the use of RLS approach in IDHP further improves the accuracy and reduces the noise of the incremental model identification. These merits make IDHP method a successful candidate for online fault-tolerant control.

5.4.2. Online Fault-Tolerant Control

In practical cases, system uncertainties, such as a time-varying component in the system, unexpected changes due to failures and measurement disturbances, need to be taken into account [26, 66, 110, 111]. However, conventional DHP methods usually need an off-line learning stage to train the global system model. The reason is that the slow online learning capability of DHP may lead to a large overshoot and lost control at the initial stage. This feature makes the DHP unable to perform Fault-Tolerant Control (FTC) in many applications, especially when the system changes suddenly and might become unstable. On the other hand, IDHP method can online identify the incremental model faster and more accurately. This merit makes IDHP method also suitable for fault-tolerant control tasks with sudden changes in the system.

Comparison of the Performance of IDHP and DHP

This chapter validates the IDHP method in 3 different situations that sudden changes are introduced to the original plant dynamics: 1) the changes in signs of the C_{z1} and C_{m1} terms in Eq.(5.35), 2) the changes in signs of the b_2 and b_4 terms in Eq.(5.35), and 3) the changes in signs of all the 4 terms. In these situations, the conventional DHP are unable to change the policy quick enough before the divergence of the states even with a perfect Fault Detection and Isolation (FDI). The sudden changes introduced may lead to unstable open loop plant. The policy trained with the original system may even increase the instability of the closed loop plant. Therefore, the actor weights will be reset to small, random numbers from the range $(-0.01, 0.01)$ when the fault is detected.

Figure 5.8 to 5.10 compare the online adaptability of the DHP method and the IDHP method in the presence of the aforementioned three different situations with sudden changes in the system dynamical model. These changes are introduced after the convergence of the policy for the original system. Because the DHP method can only deal with online tracking problems with initial states within the range $\pm 4°$, the subfigures introduce the sudden changes in three different angle of attack values of the reference signal: (a) $\alpha^{ref} = -4°$, (b) $\alpha^{ref} = 0°$, and (c) $\alpha^{ref} = 4°$.

When the changes are introduced at different, random times in these FTC tasks, the IDHP method has a success rate of 91.1% and an average tracking error of 0.067 degrees. However, the average settling time after the changes is 0.43 seconds, which is slower than the online tracking problem with the original system as depicted in table 5.1. The main reasons may be as follows: 1) the changes of the dynamical model may make the system unstable; 2) the fault detection also needs time; and 3) the RLS approach may take longer to identify the system from a fault estimation than from an initialized uninformative estimation to the same accuracy.

In addition to these performance results, different levels of simulation data are still needed for a better understanding of the success of the IDHP in FTC tasks. The rest

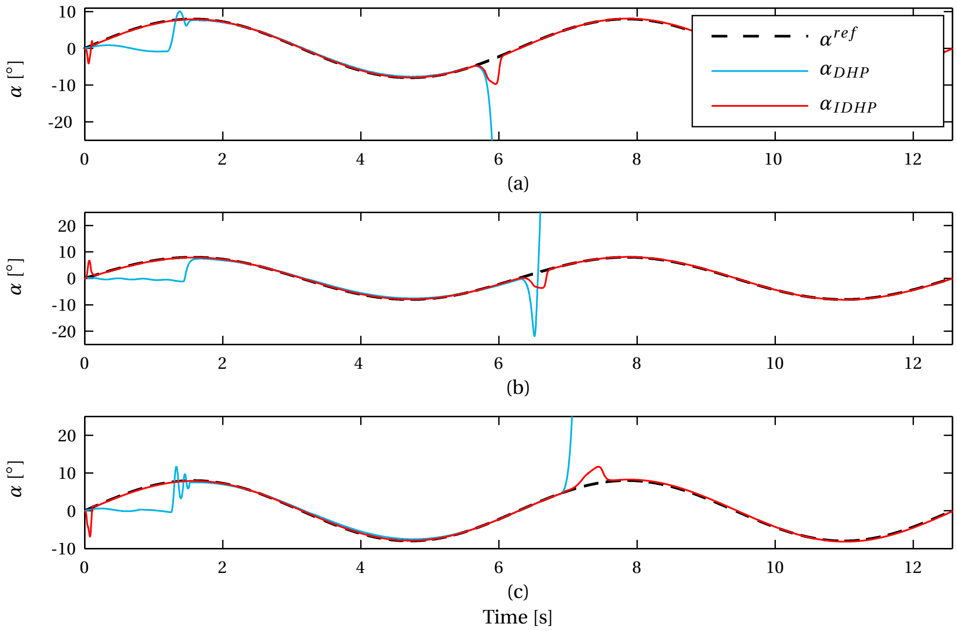


Figure 5.8: Online fault-tolerant control using DHP and IDHP approaches in the presence of sudden changes in signs of C_{z1} and C_{m1} at 3 different angle of attack values.

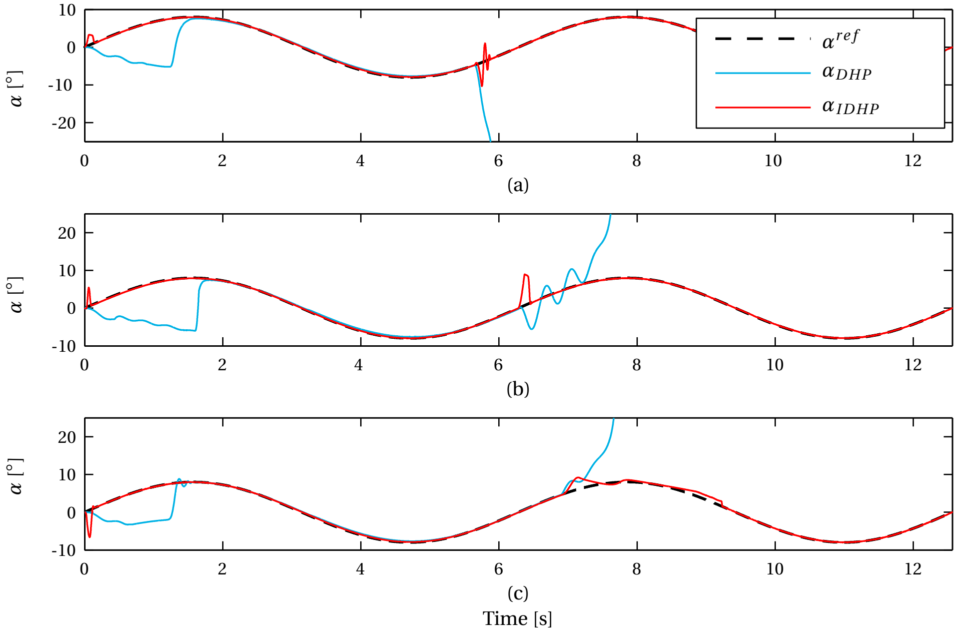


Figure 5.9: Online fault-tolerant control using DHP and IDHP approaches in the presence of sudden changes in signs of b_2 and b_4 at 3 different angle of attack values.

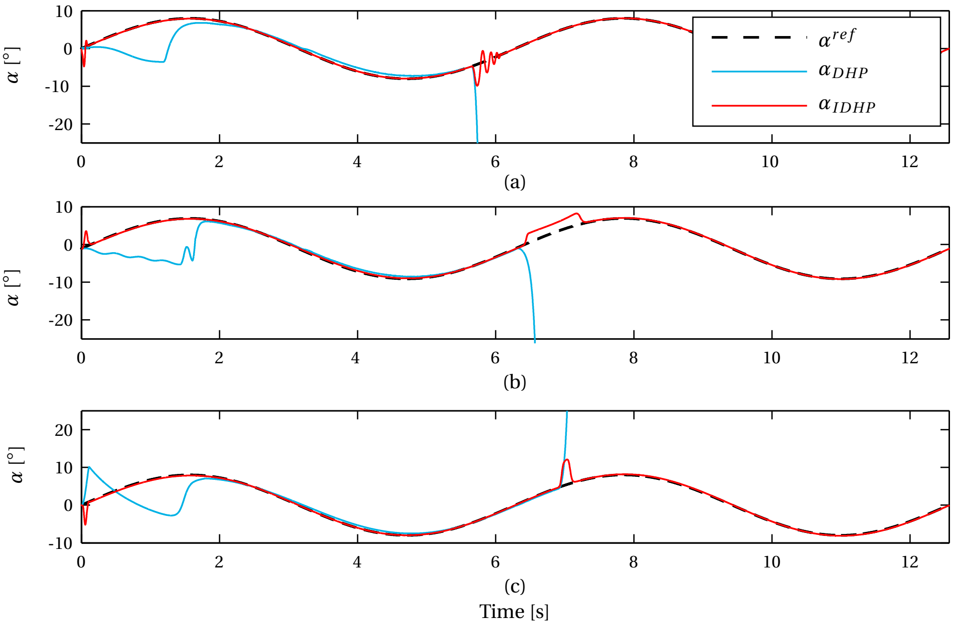


Figure 5.10: Online fault-tolerant control using DHP and IDHP approaches in the presence of sudden changes in signs of C_{z1}, C_{m1}, b_2 and b_4 at 3 different angle of attack values.

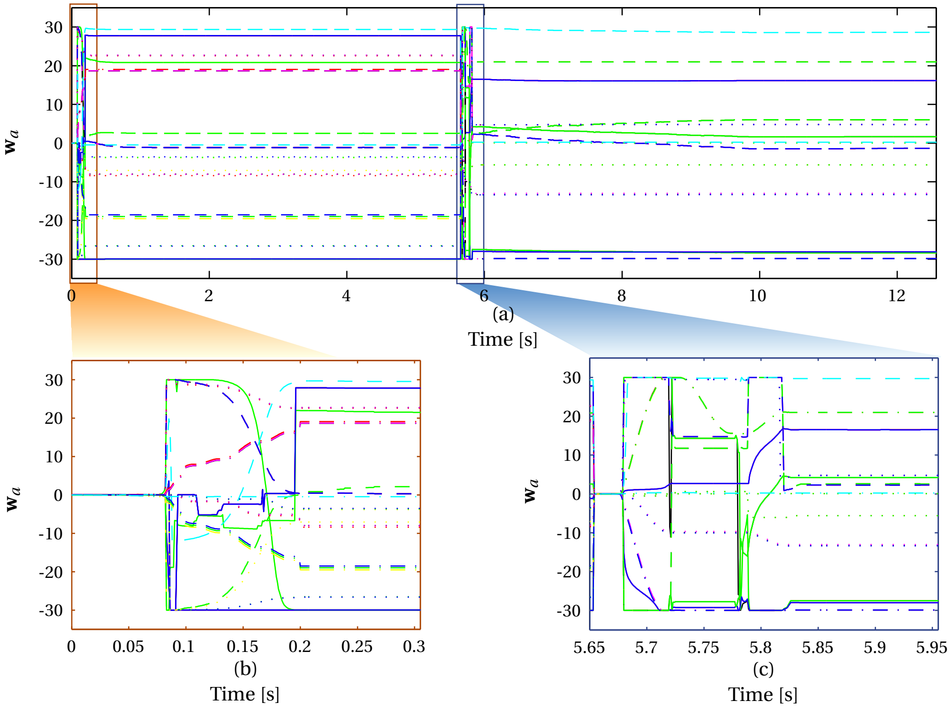


Figure 5.11: Convergence of the actor weights using the IDHP approach.

of results are based on the online FTC task in the presence of the third situation when $\alpha^{ref} = -4°$, i.e., Fig. 5.10 (a). Figure 5.11 illustrates the adaptation of the 38 actor weights during this online FTC task. This figure demonstrates the convergence of the actor weights, which represents the achieved control policy. Figure 5.11 (b) and (c) take a closer look at the first 0.3 seconds at the beginning and after the changes introduced, respectively.

Fault Detection Using the Innovation Term

The aim of this section is to validate the adaptability of the IDHP method to uncertainties. Although there are many different, elaborate approaches to FDI [66, 111, 112] , this chapter uses a simple method to detect the fault for IDHP and assumes a perfect FDI for conventional DHP (i.e., the controller knows when the system will change and reset the actor weights at that moment). These presented results indicate that conventional DHP method cannot perform FTC tasks especially when the system becomes unstable after sudden changes, although this method assumes a perfect FDI. In contrast, IDHP method can detect the system fault, reset the actor weights, and rapidly adapt its policy (the actor weights) online before the system states diverge.

IDHP method monitors the innovation term ϵ_t in Eq. (5.29) in RLS approach so as to detect the fault. This term is the difference between the measured next state and its one-step prediction, which reflects the accuracy of the online identified incremental model. When the system changes suddenly, the magnitude of the innovation terms will also increase and exceed a threshold, $\epsilon_{threshold}$, and trigger the fault alarm. In this case, the thresholds are chosen as $\epsilon_{threshold} = [3 \times 10^{-4}\,°,\ 0.1°/s]^T$. As depicted in Fig. 5.12, the magnitude of the innovation terms increase suddenly and immediately after the changes of the system. Although the selection of reasonable thresholds is also crucial in the FDI scope, this chapter focus on the validation of the proposed online adaptive IDHP method.

Incremental Model Identification and Prediction

The main reason for the success of the IDHP method is the use of the online identified incremental model. This model, as same as the global system model in DHP method, is used to estimate the term $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}|_m$ and $\frac{\partial \mathbf{x}_t}{\partial \mathbf{u}_{t-1}}|_m$ in Eq. (5.16) for updating the critic, and to estimate the term $\frac{\partial \hat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t}|_m$ and to predict the next state $\hat{\mathbf{x}}_{t+1}$ in Eq. (5.23) for updating the actor. Therefore, a proper control policy is based on a feasible system model. Figure 5.13 compares the one-step prediction of α obtained from the global system model, and incremental models using Ordinary Least Square (OLS) method and using RLS method during an online fault-tolerant control task. Because the DHP method fails in this task, the online identification using the artificial neural network, OLS, and RLS are identified using the same data obtained from a task using the IDHP method (the red line as shown in Fig. 5.10 (a)) for a fair comparison. Although all the predictions are feasible as depicted in Fig. 5.13 (a), the incremental models have more accurate prediction by looking at the prediction errors in Fig. 5.13 (b).

To examine the accuracy of the aforementioned partial derivative terms, Figs. 5.14 and 5.15 present these terms calculated from the global system model and incremental models. These derivative terms obtained from the incremental models (using both OLS

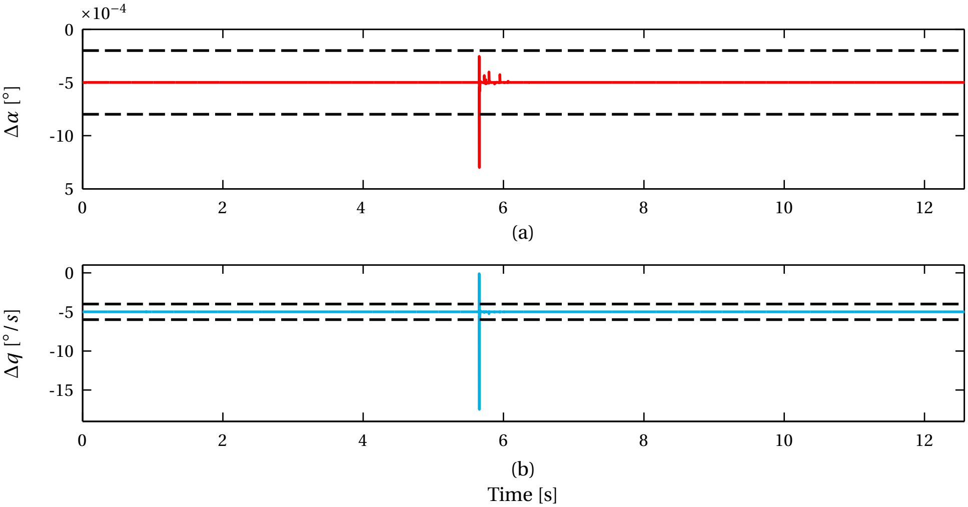


Figure 5.12: The innovation term ϵ_t obtained by RLS using IDHP approach, where the dashed lines represent the threshold.

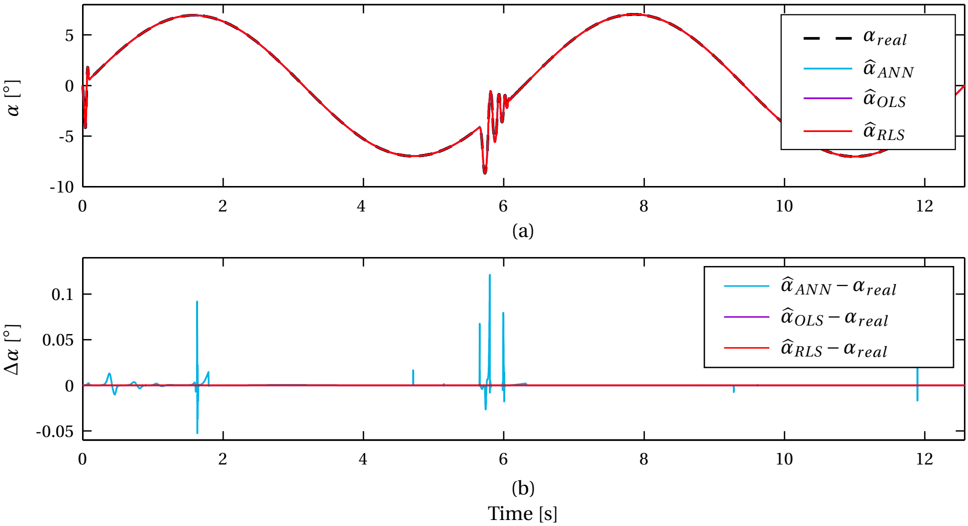


Figure 5.13: The one-step state prediction using ANN, OLS, and RLS.

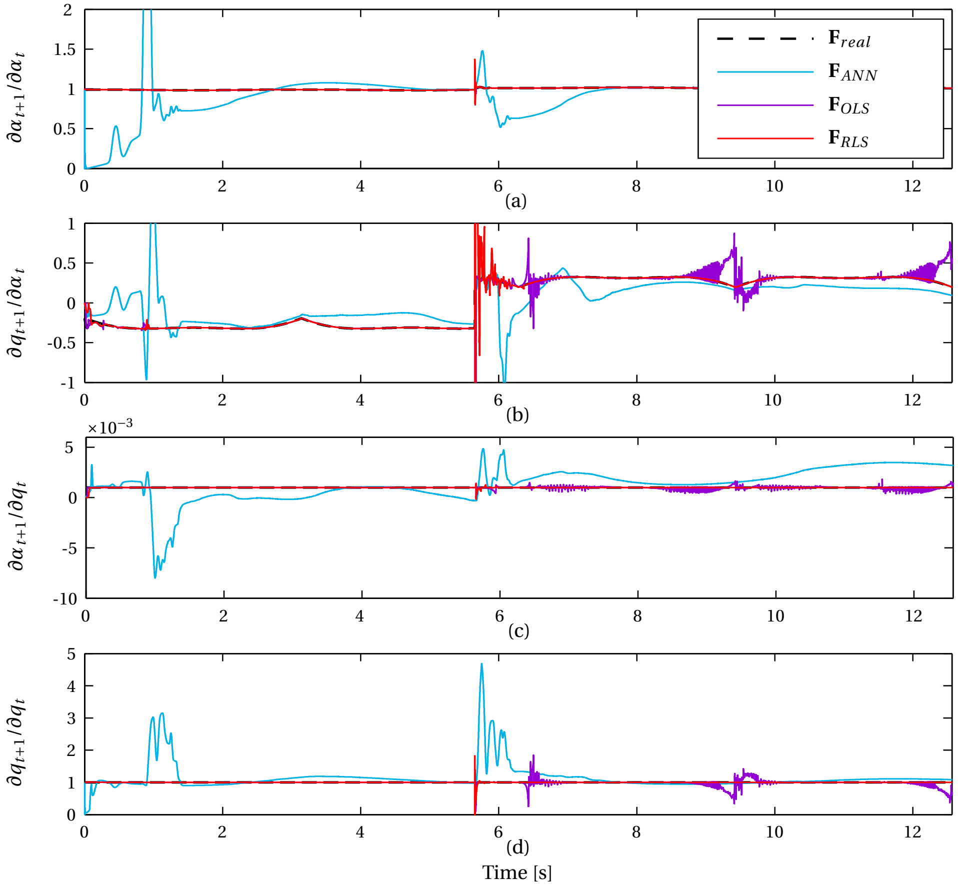


Figure 5.14: Online identified system derivatives, **F**, using ANN, OLS, and RLS.

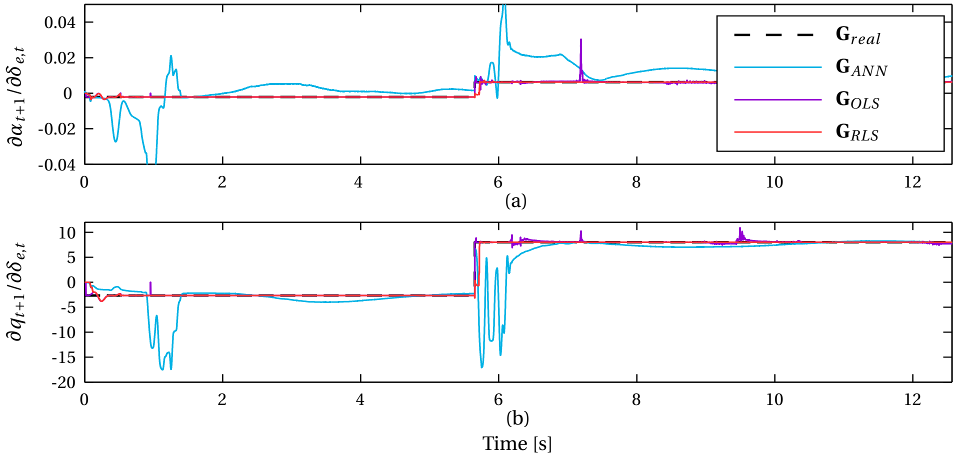


Figure 5.15: Online identified control derivatives, **G**, using ANN, OLS, and RLS.

and RLS) have higher accuracy than calculated from the neural network system model. This difference is owning to different structures and adaptation laws of the neural network model and incremental models. The artificial neural network is updated by minimizing the difference between the measurement of the state and its prediction instead of building an explicit representation for the derivatives. Therefore, it does not have to be a feasible approximation of the derivatives of the system states. Incremental model techniques, on the other hand, identify the system transition matrix and the input distribution matrix for the linearized model, which directly approximate the control derivatives of the current system states. These results elucidate the higher online adaptivity of the IDHP compared to conventional DHP methods. These figures also show that RLS outperform OLS in identifying the system derivatives.

Validation in the Presence of Measurement Noise

To further validate the robustness of the proposed IDHP method, high-frequency measurement noise is superimposed to the measurements of the states, $\check{\alpha}$ and $\check{q}$. The simulated noise is zero-mean normal distributed white noise. The standard deviations of the noise for $\check{\alpha}$ and $\check{q}$ are 0.05° and $0.005°/s$, respectively. As shown in Fig. 5.16, the performance of the fault-tolerant control task using the IDHP method is still satisfactory in the presence of the simulated noise. The settling time at the initial stage is longer because the high-frequency measurement impedes the online identification of the incremental model and slows the critic and actor adaptation, consequently. For the same reason, the tracking error becomes bigger after the sudden change of the dynamical model.

If the amplitude of the noise is further increased, the adaptation of the actor will start even later. In consequence, the states may exceed their allowable range after the sudden change, which makes the system unstable. For the tracking task without sudden changes of the dynamical model, the success rate remains at the same level (above 90%) if the standard deviations of the noise for $\check{\alpha}$ and $\check{q}$ are increased to 0.5° and $0.05°/s$, respectively. It is shown that the IDHP method is robust to measurement noise. The tolerance of the measurement noise can be higher, and the performance can be further improved, by using the adaptive least square method to identify the incremental model or using a filter to reduce the measurement noise. This is beyond the scope of this chapter and is recommended for future research.

5.5. Conclusion

This chapter develops a new approach to design an online adaptive control method that does not need a priori information of the system dynamics, called Incremental model based Dual Heuristic Programming (IDHP). This method is based on DHP methods, which are adaptive and use nonlinear function approximators, such as neural network functions. The IDHP method uses Recursive Least Square (RLS) approach to online identify the incremental model instead of the global system model. This approach, therefore, does not need the off-line training stage and can accelerate the online learning efficiently. Online learning ability is of great practical values, especially when a priori knowledge of the system is unknown or the system dynamics are changed in operations.

This chapter conducts two nonlinear flight control tasks by applying both the DHP method and the IDHP method. The first is an online reference tracking task. The results

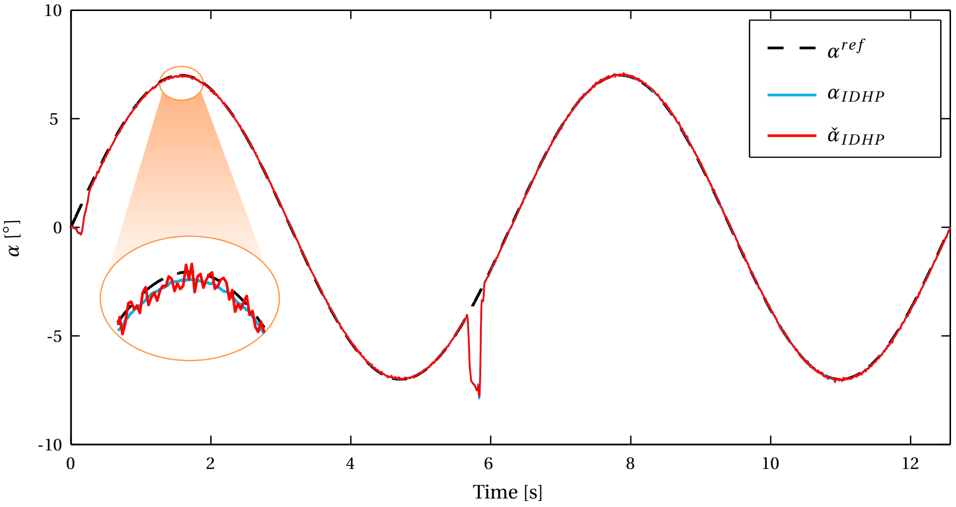


Figure 5.16: Online fault-tolerant control using IDHP approach in the presence of measurement noise.

indicate that the presented IDHP method, compared to the DHP method, can accelerate the online learning, improve the precision, and deal with a wider range of initial states. The second task is fault-tolerant control in the presence of sudden changes of the original system dynamics. The results also reveal that the IDHP method can successfully control a new and unstable system adaptively before the states diverge, while DHP fails to. Although this chapter focus on the artificial neural networks as nonlinear function approximators, the result is expected to be comparable for other global function approximators.

This study is an extension of the incremental Approximate Dynamic Programming (iADP) method, which uses a quadratic cost function, and the IHDP method, which is based on HDP. In comparison to these methods, the presented IDHP method can be used in more general control problems and shows higher learning capability, efficiency, accuracy, online adaptability, and robustness. Further investigation and experimentation are recommended into different types of function approximators and more complex, realistic applications.

III

High-level Guidance and Navigation

6

Hybrid Hierarchical Reinforcement Learning with Partial Observability

In previous chapters, incremental techniques were integrated into ADP methods to solve control problems for unknown, nonlinear systems. As guidance and navigation are also essential to an intelligent autonomous control system, this chapter develops a *hybrid* Hierarchical Reinforcement Learning (hHRL) method for high-level decision making. This method consists of several hierarchical levels, where each level uses different methods to optimize the learning with different types of information and objectives. This chapter starts with a brief introduction to Markov and semi-Markov decision processes and several RL methods, in Section 6.2. Section 6.3 describes the mobile robot system and formulates the autonomous guidance and navigation problem. After that, Section 6.4 presents the hHRL approach for online guidance and navigation and sets out rules for establishing the hierarchies and sub-tasks. The implementation and simulation results in stationary and non-stationary mazes are presented in Section 6.5. The results indicate that the proposed hHRL method can deal with multiple objectives and partial observability, possesses the ability to transfer learning, and also learns in non-stationary environments.

Autonomous guidance and navigation problems often have high-dimensional spaces, multiple objectives, and consequently a large number of states and actions, which is known as the 'curse of dimensionality'. Furthermore, systems often have partial observability instead of a perfect perception of their environment. Recent research has sought to deal with these problems by using Hierarchical Reinforcement Learning, which often uses same or similar RL methods within one application so that multiple objectives can be combined together. However, there is not a single learning method that can benefit all targets. To acquire optimal decision-making most efficiently, this chapter proposes a *hybrid* Hierarchical Reinforcement Learning method consisting of several levels, where each level uses different methods to optimize the learning with different types of information and objectives. A guidance and navigation algorithm is provided using the proposed method and applied to a multi-objective task: getting to a target area while avoiding obstacles. The navigation environments are complex, partially observable, and completely unknown at the beginning. The results indicate that the proposed hybrid hierarchical reinforcement learning method can help to accelerate learning, to alleviate the 'curse of dimensionality' in complex decision-making tasks, to naturally reduce the uncertainty or ambiguity at higher levels, to transfer the learned results within and across tasks efficiently, and to apply to non-stationary environments. This proposed method can potentially yield a near-optimal policy hierarchically for autonomous guidance and navigation without a priori knowledge of the system or the environment.

6.1. Introduction

Reinforcement learning is a collection of algorithms that learn what actions to take to affect the system state in order to maximize some numerical reward from interaction with the environment. This method links bio-inspired artificial intelligence techniques to the field of guidance, navigation and control to overcome some of the limitations and problems in conventional model-based methods requiring the system model and complete environment information. RL has been successfully applied to solve optimal decision-making problems in known or small-scaled environments [29, 31]. Nevertheless, when applied to complex multi-objective tasks, such as autonomous guidance and navigation problems, traditional RL methods are still rendered intractable by the following challenges:

- The large state and action space, which may lead to the 'curse of dimensionality'.
- The multiple, conflicting objectives.
- The transfer of learning across tasks.
- The partial observability of the system state and unknown, non-stationary environments.

Many studies have provided successful solutions to part of the aforementioned challenges. To solve the exponential growth of states caused by the complexity of the environment and system, many RL algorithms apply Approximate Dynamic Programming (ADP), which uses an approximator to approximate the cost/value function, so that they can be used to solve the optimality problems with large or continuous state spaces

[32, 34, 80, 81]. Besides the large state and action space, multiple objectives also scale up the problem complexity [113]. A considerable amount of multi-objective optimization strategies have been developed [114–118] to simultaneously solve several tasks with different reward systems. Another source of problem complexity is the partial observability, i.e., in the real world, the agent might not have a perfect perception of the states or the environment [45]. Nevertheless, the agent has to pick an action based on its current observation and its knowledge accumulated through the involvement with the environment [119]. The framework dealing with Partially Observable Markov Decision Process (POMDP) problems has been developed, mainly based on calculating beliefs, to decide how to act in these situations and still remains an active area of research [36, 48, 49, 60, 120].

Recent research attempts, dealing with the 'curse of dimensionality', focus on the hierarchical decomposition with Hierarchical Reinforcement Learning (HRL) [32, 37, 61, 121], such as HAMs[122, 123], options[124], MAXQ [125, 126], and HEXQ [127]. These methods replace state-to-action mapping by a hierarchy of temporally abstract actions which operate over several time steps. HRL is a natural approach to problem-solving, with which a complex problem can be solved by decomposing it into several smaller and simpler problems. Hierarchical decomposition speeds up learning for multi-objective tasks by allowing different objectives in different levels. HRL also naturally reduces the uncertainty and ambiguity induced by partial observability at higher levels because some hidden variables can be estimated at different levels. Some of those hierarchical learning methods, especially for large scale, partially observable environments, form a branch called Hierarchical POMDP (H-POMDP)[37, 128–130]. The hierarchical Q-learning [131] has been proposed to deal with partial observable maze navigation problems based on the HAMs design [123].

Current HRL approaches often use same or similar RL methods within one application so that multiple objectives can be combined together. Otherwise, a more careful design with expert knowledge may be needed. However, it is indicated[58, 59] that there is not a single learning method that can benefit all targets. To acquire optimal decision-making most efficiently, different levels within one HRL application often need different learning methods, learning types, rewards assignment, and state information.

Furthermore, there is less research on the explicit rules of establishing the hierarchies and of assigning the rewards. These designs, such as the components of the hierarchy or the decomposition rules and the reward signals, have to be decided in advance [61]. These designs, involving engineers' preferences, may prevent the transfer learning from one application to another even similar application.

Transfer Learning (TL), from source domain models to the target domain model, has been studied over the past decades for clustering problems [132], machine learning tasks [133–135], planning tasks [136], etc. In recent years, TL has attracted gaining attention in RL community [137–141]. TL allows for reusing some previous results, learned from several related tasks, in a subsequent task [139]. In RL, for example, policies learned in one environment can be applied or partially applied to another environment, depending on how their state/action spaces differ. The transferability of RL methods across different guidance and navigation tasks is of tremendous importance from a practical perspective.

Conventional RL studies assume that the environment is stationary within one task.

However, in the real world, guidance and navigation environments are often non-stationary. For example, when an agent episodically traverses in an office-like environment, the position of desks and chairs, the open/closed windows and doors, and walking people may change dynamically over episodes or even over time. There has been some successful adaptive learning methods [49, 142–144] and mapping approaches [145–148] dealing with non-stationary environments. Nevertheless, learning in non-stationary environments is still challenging and promising due to the increasing prevalence in practical applications [143, 145].

The aim of this chapter is to address the aforementioned knowledge gaps by proposing a *hybrid* Hierarchical Reinforcement Learning (hHRL), with explicit rules to establish the hierarchies. The hHRL method allows for different methods in different levels and sub-tasks, assimilates multiple objectives, and possesses transferable learning results. This proposed method is validated, in an online guidance and navigation task with partial observability, in respect to the following abilities:

- Learning efficiency.
- Transferability across tasks.
- Applicability in non-stationary environments.

The remainder of this chapter is structured as follows. Section 6.2 begins with a brief introduction to Markov and semi-Markov decision processes and different RL methods that are used in this chapter. Section 6.3 describes the mobile robot system and formulates the autonomous guidance and navigation problem. Then, section 6.4 presents a hybrid Hierarchical Reinforcement Learning approach for online guidance and navigation, sets out rules for establishing the hierarchies and sub-tasks, and presents the implementation in each sub-task. The proposed method and implementations are validated in the guidance and navigation task with partial observability in section 6.5. The last section concludes the advantages and disadvantages of using the proposed method and addresses the challenges and possibilities of the future research.

6.2. Foundations

This section introduces the fundamentals of Hierarchical Reinforcement Learning, including Markov Decision Processes (MDPs) and Semi-Markov Decision Processes (SMDPs), and several RL methods that will be used in this chapter.

6.2.1. Markov Decision Processes and Semi-Markov Decision Processes

Markov Decision Processes describe the mathematical framework of decision making that Reinforcement Learning aims to solve [29, 31, 149]. An MDP consists of 5 elements:

1) a set of states $s \in \mathscr{S}$,

2) a set of actions $a \in \mathscr{A}$,

3) the transition probabilities $\mathscr{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$,

4) the expected immediate reward $\mathscr{R}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$, and

5) the discounted rate $\gamma \in [0,1]$,
where r_t denotes the immediate reward of the next state s' after taking a possible action a. This 5-tuple specifies the major aspects of the dynamics of an MDP.

With the concept of an MDP dynamical model, the value function under policy π is defined by the Bellman equation [31]. RL approaches seek to improve the policy by approximating and maximizing the value function until an optimal policy π^* is ultimately established. The state value function, $V^\pi(s)$, is defined as the expected return starting from s and following the current policy π. The corresponding Bellman optimality equation for $V^*(s)$ is shown as follows [31]:

$$\begin{aligned} V^*(s) &= \max\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\right\} \\ &= \max_{a\in\mathscr{A}} E_{\pi^*}\left\{r_{t+1}+\gamma V^*(s')|s_t = s, a_t = a\right\}. \end{aligned} \tag{6.1}$$

Similarly, the Bellman optimality equation for the state-action value function, $Q^*(s,a)$, is defined as the expected return starting from s, taking the action a, and thereafter following the optimal policy π^*:

$$\begin{aligned} Q^*(s,a) &= E\{r_{t+1}+\gamma \max_{a'\in\mathscr{A}_{s'}} Q^*(s',a')|s_t = s, a_t = a\} \\ &= \mathscr{R}^a_{ss'}+\gamma\sum_{s'}\mathscr{P}^a_{ss'}\max_{a'\in\mathscr{A}_{s'}} Q^*(s',a'). \end{aligned} \tag{6.2}$$

An MDP defines only the sequential decision process, while an SMDP, which is a generalization of the MDP, also takes the time between one decision and the next into account [61, 124]. The SMDP is the basis of HRL. If action a is executed in the current state s_t, the next decision will be made when the next state s_{t+1} occurs after certain time τ. Its joint transition probabilities is written as $P(\tau, s_{t+\tau} = s'|s_t = s, a_t = a)$ [61]. The remaining or waiting time in the current stage, τ, can be either real valued in continuous-time discrete-event systems or integer valued in discrete-time systems as in this chapter. A common way of writing the Bellman optimality equation of SMDPs [61] for state value function, V^*, is

$$V^*(s) = \max_{a\in\mathscr{A}}\left\{\mathscr{R}^a_{ss'}+\sum_{(s',\tau)}\gamma^\tau P(\tau, s_{t+\tau} = s'|s_t = s, a_t = a)V^*(s')\right\}, \tag{6.3}$$

and for state-action value function, Q^*, it is

$$Q^*(s,a) = \mathscr{R}^a_{ss'}+\sum_{(s',\tau)}\gamma^\tau P(\tau, s_{t+\tau} = s'|s_t = s, a_t = a)\max_{a'\in\mathscr{A}_{s'}} Q^*(s',a'), \tag{6.4}$$

where $\mathscr{R}^a_{ss'}$ is the accumulated reward within the time span $(t, t+\tau]$.

6.2.2. REINFORCEMENT LEARNING METHODS

Fundamental RL methods can be classified into three categories [21, 31]: 1) Dynamic Programming (DP), which uses a complete knowledge of the system and/or environment to compute the optimal policy; 2) Monte Carlo (MC), which does not require a priori knowledge of the environment but only sample experiences and final rewards for episodic tasks; and 3) Temperal-Difference (TD), which is a combination of DP and MC ideas and updates the policy based on the sample experience at each step. Although different, some of these ideas are joint to extensively utilize the resources and to achieve the best solution in solving the problems in this chapter.

Monte Carlo

Monte Carlo methods do not need a model of the system or environment. Instead, they execute the current policy during an iteration and update the policy afterward based on the trace of the experienced states, actions, and received rewards. This chapter uses an MC method in the higher level to estimate the state-action values $Q(s_k, a_k)$, where k represents the high-level state-action sequences. These values can be updated based on the averaged rewards, finite horizon, or discounted rewards.

This chapter focuses on the discounted rewards because they are more suitable for guidance tasks that receive a (constant) final reward r_T, if and only if it reaches the target position. After the termination of the jth iteration, the values of experienced state-action pairs, (s_k, a_k), are updated, as follows:

$$Q_{j+1}(s_k, a_k) = \gamma^{T-1-k} \cdot r_T + \max_{a_{k+1} \in \mathscr{A}_{s_{k+1}}} Q_j(s_{k+1}, a_{k+1}), \tag{6.5}$$

where $k = 1, ..., T-1$.

Q-learning

As an off-policy TD method, Q-learning will be used for online adaptation in this chapter when immediate rewards are available. This method approximates the optimal state-action value function Q^* of the *estimation policy*, which is independent of the *behaviour policy* [31]. This separation allows exploration with the behaviour policy and simplifies the analysis of the algorithm. This chapter uses a one-step Q-learning algorithm to update state-action values in discrete form with immediate rewards r_{t+1}, as follows [150]:

$$Q_{t+1}(s_t, a_t) = (1-\delta)Q_t(s_t, a_t) + \delta \left[r_{t+1} + \gamma \max_{a_{t+1} \in \mathscr{A}_{s_{t+1}}} Q_t(s_{t+1}, a_{t+1}) \right], \tag{6.6}$$

where δ is a *learning-rate parameter.*

When applied to SMDP, Q-learning updates the value function when it receives an immediate reward. This immediate reward $r_{t+\tau}$ is assigned after the agent executes action a in state s and gets into another state s' after time τ. In discrete-time systems, Q-learning updates the estimated state-action value $Q_k(s, a)$ of the optimal $Q^*(s, a)$ as follows [61]:

$$Q_{k+1}(s, a) = (1-\delta)Q_k(s, a) + \delta \left[R_{t,\tau} + \gamma^{\tau} \cdot \max_{a' \in \mathscr{A}_{s'}} Q_k(s', a') \right]. \tag{6.7}$$

where $R_{t,\tau}$ is the accumulated reward during the waiting time:

$$R_{t,\tau} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{\tau-1} r_{t+\tau}. \tag{6.8}$$

Q-learning is adopted as the 'flat' method, which refers to non-hierarchical RL, in this chapter because it has three advantages [30, 31, 61, 151]. First, it uses iterative data samples obtained from the real world test or from produced stochastic simulations, rather than access to the explicit knowledge of the expected rewards or the state-transition probabilities. This trait allows Q-learning to be used as a model-free RL method. Second, it uses state-action values instead of state values. Thus, in discrete-event systems, finding optimal actions does not require one-step ahead search or access to the one-step action models. Third, it is easy and explicit to store all state-action values for small-scaled problems. And it is possible to extend this advantage to large-scale problems by using function approximation methods or HRL methods.

6.3. AUTONOMOUS GUIDANCE AND NAVIGATION TASK

This section will introduce an autonomous guidance and navigation task in unknown mazes with multiple objectives and partial observability.

6.3.1. SYSTEM DESCRIPTION

As depicted in Fig. 6.1, the agent represents a flying robot navigating in indoor environments with limited sensors. It does not know its *absolute state*, i.e. the exact position in the environment, but only observes a *relative state*, $\mathbf{s} \in \mathcal{S}$: 3-step short-sight (obstacle in the 1, 2, 3-step away or nothing) in 3 directions (front, left, and right), represented by $s_f \in \mathcal{S}_f$, $s_l \in \mathcal{S}_l$, and $s_r \in \mathcal{S}_r$, and its heading angle $s_h \in \mathcal{S}_h : \{0°, 90°, 180°, 270°\}$ representing {North, West, South, East}. Thus, this robot only has 256 possible *observed states* $\mathcal{S} : \mathcal{S}_f \times \mathcal{S}_l \times \mathcal{S}_r \times \mathcal{S}_h$. If the absolute state is fully and always observable, this problem is deterministic. On the other hand, if the system only senses its relative state in the environment, this problem becomes stochastic. The observed relative states can be ambiguous in complex environments, which is also referred to as the POMDP. Figure 6.2, as an example, presents two situations, where the agent senses the same relative state $\mathbf{s} = [1, 0, 0, 180°]$. They are, however, in two different absolute states and should take different actions to approach the target area. Furthermore, if the agent takes the same action in these two situations, it will sense different next states. This partial observability will prevent the state-action values from converging with a 'flat' RL method.

The agent has limited mobility $\mathcal{A}$: {turn left, turn right, move forward}. If the agent moves towards an obstacle when it is next to the obstacle, it will automatically turn around and result in a penalty '-4'. When it reaches the target area, the robot will land at this area and get a final reward '10'. This agent does not know the map of this maze in advance, but has an internal memory of its trajectory, which will be removed before another episode, and a long-term external memory, which can only store a map. This chapter focuses on the high-level online guidance and navigation rather than the system model or the low-level control and, therefore, makes the following assumption:

***Assumption 1**. The flying robot is able to accurately perform the one-step actions in $\mathcal{A}$.*

6.3.2. PROBLEM DESCRIPTION

Figure 6.3 shows an indoor guidance and navigation problem for the agent in a discrete maze. This *maze A* is a benchmark maze from Parr [123], which has about 3600 possible

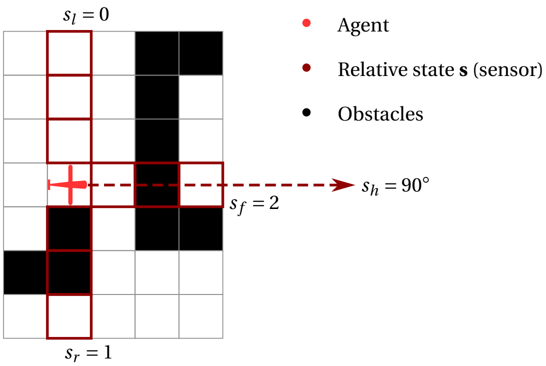


Figure 6.1: An example of the agent, sensing a relative state, $\mathbf{s} = [s_f, s_l, s_r, s_h]$, in a discrete environment.

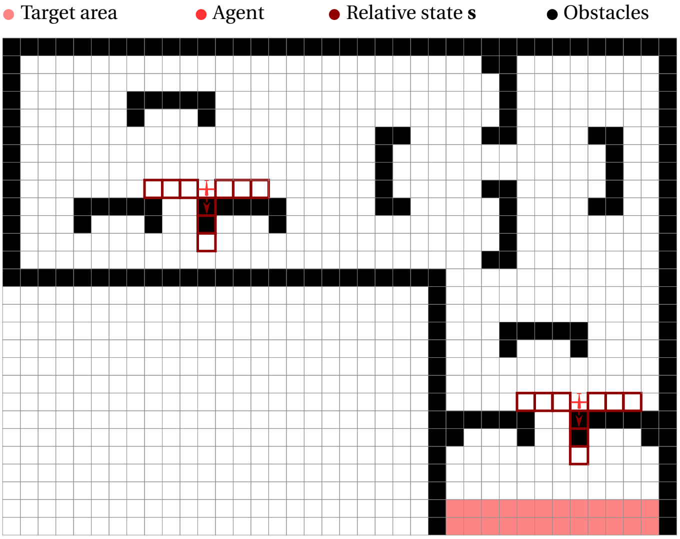


Figure 6.2: An example of an agent in two possible situations, sensing the same relative state $\mathbf{s} = [1, 0, 0, 180°]$, which, however, are in two different absolute states (positions) and should take different actions to approach the target area.

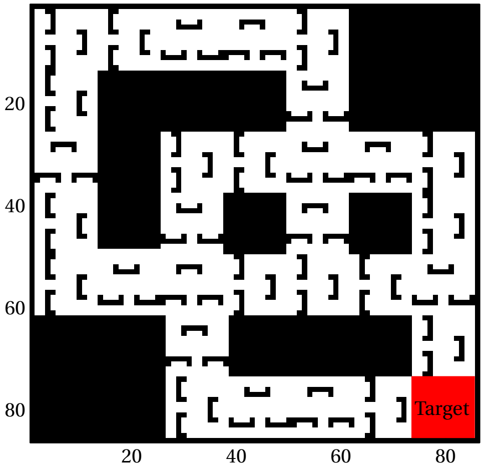


Figure 6.3: A guidance and navigation problem with obstacles and a target in *maze A*.

absolute states (without considering the heading) and a target area. The agent has to perform two tasks at the same time: avoiding the obstacles, and approaching the target area as soon as possible.

Compared to Parr's original maze problem [123], the current task is more complex and realistic with partial observability, unknown starting positions, limited mobility, and a possibly changing environment. The agent does not have any a priori information about the environment or its position and learns everything online. This maze environment and the target position may change, and the agent needs to transfer the learning result across different environments. In addition, some of the obstacles in this maze can also be non-stationary.

6.4. HYBRID HIERARCHICAL REINFORCEMENT LEARNING

This section presents a hybrid Hierarchical Reinforcement Learning (hHRL) method to allow for different methods and types of state information in each level and each subtasks.

6.4.1. DECOMPOSITION AND HIERARCHIES

For online guidance and navigation in a priori unknown environments, decomposition of tasks and abstraction of actions allow agents to solve current sub-problems and to ignore irrelevant details at the current level. Each higher level uses a partial description of the environment, which can partition the environment into *sub-environments*, from a top-down point of view, or *macro states*, from a bottom-up perspective. This feature may naturally reduce the uncertainty or ambiguity induced by partial observability of the environment. The activities or decisions in higher levels are called *macro actions* or *behaviors*, which instruct the policy in the lower level. The original one-step actions for each state are called *primitive actions*.

In guidance and navigation problems, the environment can be decomposed into several sub-environments based on physical isolation, such as rooms or floors. For more general environments, such as *maze A*, which does not have clear, observable isolation, it can be decomposed, based on the physical distance, into macro states, e.g., encompassing 12 × 12 absolute micro states as in Fig. 6.4, which results in coarser-grained behavior $b \in \mathscr{B}$: {go North, go West, go South, go East}. Therefore, the two situations in Fig. 6.2 can be distinguished by the macro states and follow different behaviors: go West and go South.

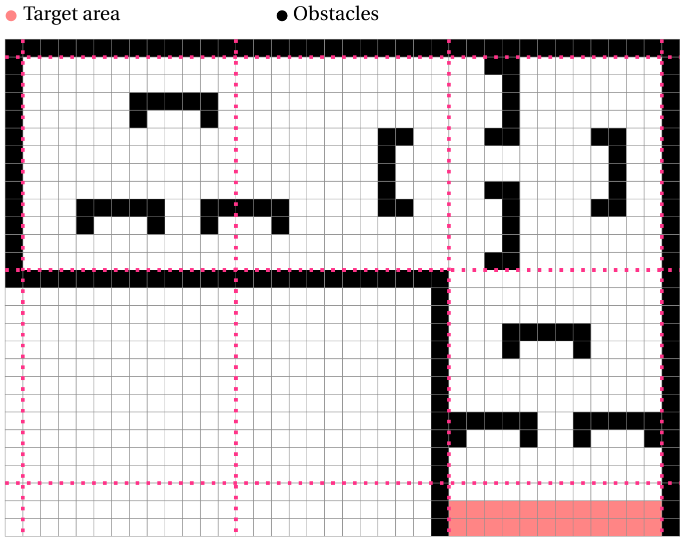


Figure 6.4: An example of environment decomposition. The pink dotted line decomposes the environment into macro states, each of which encompasses 12 × 12 absolute micro states.

The maze guidance and navigation problem can be addressed with a hierarchical structure as depicted in Fig. 6.5, using the following rules:

- ***Rule 1***. *The hierarchical levels depend on the state abstraction levels: the more abstracted, the higher the level.*
- ***Rule 2***. *Each sub-task only focuses on one objective, uses its related reward assignment system, and has a corresponding value function.*
- ***Rule 3***. *If there are multiple sub-tasks within one level, an order of priority is necessary to be determined for evaluating the importance of the related value function. The order of priorities may depend on the inherent properties or can change based on a higher level instruction.*

In Fig. 6.5, the agent operates in Level 3 (L3) on micro states, in Level 2 (L2) on macro states, and in Level 1 (L1) based on the cognition of the entire situation. Except for the decomposition into hierarchical levels, the tasks in some levels need to be further decomposed into sub-tasks, each of which focuses on only one objective and has one value function to store the learned result. In Level 2, the 1st sub-Task (L2.T1) attempts to expand the explored area in the maze for a long-term interest with multiple iterations, and

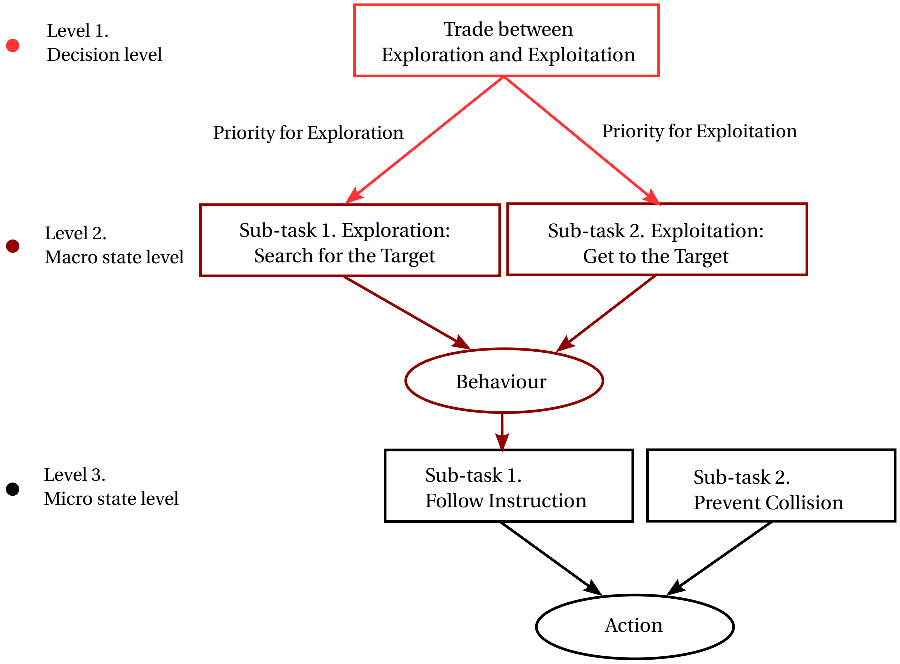


Figure 6.5: An architecture for hybrid Hierarchical Reinforcement Learning.

the 2nd sub-Task (L2.T2) focuses on the exploitation of the greedy policy to approach the target area in the current iteration. In Level 3, the 1st sub-Task (L3.T1) is following the higher level instruction, which is either exploration or exploitation, to get to the target area, while the 2nd sub-Task (L3.T2) has another objective: preventing collision with the wall or obstacles. These sub-tasks in the same level are decomposed for ease of changing the priority and of transferring the learned results to different tasks. If the maze has been thoroughly explored, L1 will give priority to L2.T2, and, on the other hand, when the maze is unexplored, or when the environment changes, L1 may decide to give priority to L2.T1 to better estimate the state and to find the target. In the 3rd level, L3.T2 always takes priory over L3.T1 in this aerial vehicle guidance and navigation task for safety reason.

6.4.2. HYBRID LEARNING

There is no single method that can benefit all targets [58, 59]. This is also applicable to HRL. Within a hierarchical structure, the state and action spaces in different levels can be discrete, continuous, or even hybrid. For instance, the lower level has a continuous state-action space, where approximate dynamic programming method might be more useful, while abstracted higher levels may have discrete state and action spaces, where Q-learning and SARSA may be more efficient. Different sub-tasks within the same level can also have different learning requirements with different reward assignment systems, such as a final reward or immediate rewards.

Therefore, the hHRL method is proposed to allow for different methods and different types of state information in each level or even in each sub-task, adhering to the following rule:

- ***Rule 4**. Within one level, different sub-tasks are allowed to use different learning methods, state information, learning types, reward assignment systems, as long as the learned result can be transferred into a same-structured value function.*

The aim of the hHRL method is to better use the acquired information and appropriate methods to effectively tackle each sub-task.

Relative and Absolute States

In this guidance and navigation problem, the agent only has partial observability with a relative state **s**. This relative state has only 256 possibilities as depicted in section 6.3. On the other hand, *maze A*, as an example, has about 14400 possible absolute states (3600 × 4, considering the heading), and this amount can ever-increase due to the increasingly expanded environment. Although relative state brings ambiguity into maze problems, it also, to some extent, limits the growing of the state space and consumes less computational and memory resources. In addition, the relative state information can directly meet the objective of preventing collisions and remains flexible in uncertain environments. Therefore, the relative state is still used in the lower level (L3) in the proposed hHRL method.

Absolute position has its own benefit in target approaching tasks, especially for small scaled problems. The higher level (L2) is operated based on macro states, each of which is a collection of nearby absolute micro states. These macro states are absolute positions in the maze, of which there is a relatively small amount. The higher level estimates the absolute macro state of the agent and focuses on the objective of approaching the target.

Reward Assignment System

External rewards, used in most RL algorithms, are positive or negative outcomes received from the external reward system or other environmental sources. They are tangible, such as consuming energy or reaching a charger in a robot navigation task. The external rewards of this guidance and navigation task are the immediate obstacle collision penalty, $r_t = -4$, and the final reward, $r_T = 10$, as depicted in section 6.3.1.

Besides external rewards, HRL algorithms also require internal rewards, which are intangible and come from the sense of performance itself, such as accomplishment of following a higher level instruction or achievement of a sub-task. The internal reward can either be a reward observed immediately after taking an action in the current state, or be a delayed reward returned after termination of a higher level action. Assigning suitable internal rewards can be difficult. It can affect the evaluation of the actions in each state following an instruction.

In this maze guidance and navigation problem, two internal rewards are assigned. First, if there is an obstacle in front of agent indicated by s_f, the agent will receive an immediate penalty according to the distance from the obstacle: $r_t = -2$ if $s_f = 1$, $r_t = -1$ if $s_f = 2$, and $r_t = -0.5$ if $s_f = 3$. This internal reward assignment is a supplement to the collision penalty. Second, if the agent in a macro state successfully follows the behavior from L2 and enters the next macro state at time τ, it will obtain a delayed reward $r_\tau = 2$.

Localization with Mapping

This chapter deals with POMDP problems by using hierarchical state information instead of estimating the absolute state (also known as belief state) in the lower level. Nevertheless, to approach the target area, the agent still needs to localize itself in the map in a higher level. In some practical POMDP tasks, the agent may have an accurate relative state as in this chapter and an inaccurate absolute positioning. This additional positioning information can be used to estimate its macro state. In a static environment, positioning can also be realized by using the mapping strategy.

The agent has an internal memory of its trajectory, which can be used for localization in the current episode and also for drawing a map to ease future tasks. The agent explores in the maze and can simultaneously draw a map based on its experienced micro states and actions. Once this episode finishes, the incomplete map will be compared to and then used to enrich the map in the external memory. The off-line learning phase, e.g., with Monte Carlo, can use this map together with its trace in the current episode to evaluate and update value functions. From the second episode onwards, the agent can create a local map online and match itself to this map to estimate its absolute macro state in the maze.

A partial map, which only contains the static environment, such as walls, is more efficient in practical use because surrounding obstacles are not always stationary. To achieve this partial map, the sensors are modified to measure if the obstacles are stationary, such as walls, or non-stationary, such as chairs or tables. With this information, the proposed hHRL method can be efficiently used for non-stationary environments, which will be explained in section 6.5.3.

6.4.3. Strategy Connecting Hierarchies and Sub-tasks

The hHRL method allows for different-structured value functions for different hierarchies. These hierarchies are not connected with each other by their value functions. Instead, the higher level passes on an instruction to the lower level. This instruction can be a behavior, e.g., from L2 to L3, a list of priorities, or an activation signal for a lower sub-task, e.g., from L1 to L2 in Fig. 6.5.

The hHRL method, according to ***Rule 4***, allows for different learning methods within one level as long as the learned results are in same-structured value functions so as to be summed up to decide what actions to take. For example, L3 uses state-action value functions $Q_{L3,Ti,Pj}(\mathbf{s},a|b)$, where $(L3,Ti,Pj)$ indicates this value function is for the 3rd level, ith sub-task, and jth order in the list of priorities. These value functions can be normalized into $N_i \cdot Q_{L3,Ti,Pj}(\mathbf{s},a|b)$ with normalization factor N_i. When getting into a relative state $\mathbf{s}_t$ under behavior b_t, the agent will calculate the Q-values for possible actions by summing up the sub-task Q-values as follows:

$$Q(\mathbf{s}_t,a) = w_2 \cdot N_1 \cdot Q_{L3,T1,P2}(\mathbf{s}_t,a|b_t) + w_1 \cdot N_2 \cdot Q_{L3,T2,P1}(\mathbf{s}_t,a), \quad a \in \mathscr{A}_{\mathbf{s}_t}, \tag{6.9}$$

where the weight $w_j \geq 0$ indicates the importance of Q-values according to the order of priority P_j. These weights can be assigned, in principle, as $w_1 > w_2 > \cdots$. If a sub-task has strict priority to others in one level, e.g., preventing collisions is strictly prior to other behaviors, a simple design principle for their importance weights is provided as follows:

- ***Rule 5***. *If the order of priority, indicated by j, is strict, the weight w_j is assigned as $w_j = 2^{-(j-1)}$. If the priorities do not change, this importance assignment can be done, practically, by assigning the reward at a* 50% *difference.*

This reward assignment principle works well for L3 in these guidance and navigation problems.

This method estimates value functions for each sub-task independently, which eases the transfer of learning to other guidance and navigation tasks and other systems. In addition, this method does not sum up the complete value functions within the whole state space or state-action space, which may vary among different methods, but only the values for the current possible actions within the action sub-space as in Eq. (6.9). This trait makes the hHRL method allows for different methods in sub-tasks within the same level.

6.4.4. Implementation: Value Functions Adaptation

The concept of hHRL and its rules have been explained. This section will present the detailed implementation of the proposed hHRL in this online guidance and navigation problem, focusing on the value functions and their adaptation in each sub-task in Fig. 6.5.

Level 1

The decision level, L1, judges if the agent should focus on exploration or exploitation. This is based on whether the agent has experienced enough in an episode to localize itself, and whether the experienced environment in the current episode matches the map in its external memory. When the agent starts an episode, it only observes its surrounding environment, and L1 will give priority to exploration. Once the macro state is estimated, L1 will change the priority to exploitation. If the environment of the current episode is different from the memorized map, e.g., the maze is expanded, or the target position is changed, the exploration will again take priority.

Level 2 sub-task 1

In the second level, L2, the macro state $S \in \mathscr{S}_M$ is an absolute position, which is estimated online and may be inaccurate. Behavior b is defined over its input set $\mathscr{B}$, macro states S, the current high-level policy from L1, and additionally, over a termination condition $\beta_{stop} : \{0,1\}$. The current behavior will be terminated when

1) the agent enters into another macro state,
2) the agent stays in the current macro state for too long, and
3) the higher-level policy changes.

The first sub-task, L2.T1, is based on a look-up table, $Count(S)$, counting visits to each macro state. Note that the agent is initially not aware of its absolute state in each episode. The macro states are partitioned and marked on the external map, according to the memorized map of its first episode. At the start of the following episodes, the macro states are partitioned according to the first position in the current episode, which may be different from the partition in the external map. Once localized, the macro-state count will be transferred into the external map and used for the remainder of the current episode.

The number of visits can be seen as penalties for this sub-task L2.T1, and can be transferred into a state value function format:

$$V_{L2,T1}(S) = -Count(S) \;/ \sum_{S \in \mathscr{S}_M} Count(S). \tag{6.10}$$

The state value function can also be transferred into a state-action value function with an ideal agent model of macro states S and behaviors b. In this implementation, the instruction from L1 is an activation signal. Therefore, when L2.T1 is activated, $V_{L2,T1}(S)$ can directly decide the next behavior.

Level 2 sub-task 2

This sub-task is learned off-line with a Monte Carlo method. When an episode ends, the agent will receive a final reward r_T if it reaches the target to update the value function. This adaptation is based on the trace-back of the experienced macro-state level state-action pairs (S_k, b_k), where k represents the macro-state level state-action sequences. The MC off-line learning allows for backwards adaptation of these values, $k = T-1, T-2, ...,$ as follows:

$$Q_{L2,T2}(S_k, b_k) = \gamma^{T-1-k} \cdot r_T + \max_{b_{k+1} \in \mathscr{B}_{S_{k+1}}} Q_{L2,T2}(S_{k+1}, b_{k+1}). \tag{6.11}$$

Therefore, the final reward can be propagated backwards more efficiently than using Q-learning [131].

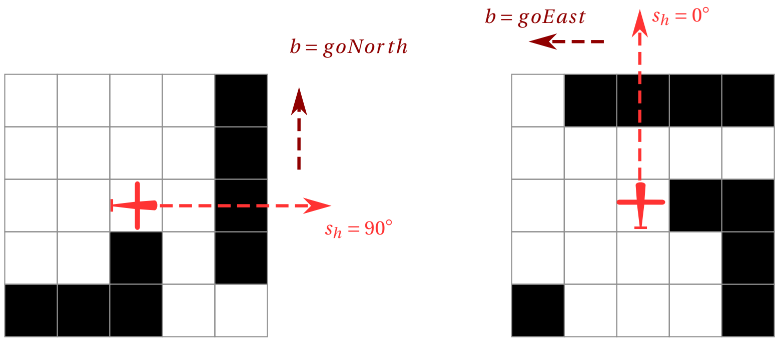
Level 3 sub-task 1

In the third level, L3, the value functions are conditional under b, for the first sub-task, which can be represented as $Q_{L3,T1}(\mathbf{s}, a \,|\, b)$. The value function dimension will be, therefore, expanded to $\mathscr{S} \times \mathscr{A} \times \mathscr{B}$, where $\mathscr{S} : \mathscr{S}_f \times \mathscr{S}_l \times \mathscr{S}_r \times \mathscr{S}_h$ [131]. To use the sample data and storage more efficiently, the state information $\mathbf{s} = [s_f, s_l, s_r, s_h]$ will be separately used in the two sub-tasks in L3.

The first sub-task, L3.T1, is following the higher level behavior b, which is actually a connection between L2 and L3. To improve the storage efficiency, this implementation first combines the heading angle $s_h \in \mathscr{S}_h$ and the behavior b and formulates a relative instruction, which is a turning angle, $\widetilde{b} \in \widetilde{\mathscr{B}} : \{0°, 90°, 180°, 270°\}$, without taking its current orientation into account. Figure 6.6, as an example, describes two situations where the agent has different heading angles s_h and behaviors b, but actually should follow the same relative instruction $\widetilde{b}$.

In addition, the obstacle prevention can be taken care of by L3.T2, which even has higher priority. Therefore, L3.T1 does not necessarily consider the obstacle-related measurements: s_f, s_l, or s_r. The value function can be reduced to $Q_{L3,T1}(\widetilde{b}, a)$, whose dimension is $\widetilde{\mathscr{B}} \times \mathscr{A}$. Note that the relative instruction $\widetilde{b}$ is processed information from the current heading angle s_h and behavior b, and can change micro-step-wise.

When the higher-level behavior b_k is terminated, the agent may receive an internal reward r_τ at time τ if the agent correctly followed the behavior. Within this time span of executing b_k, the experienced state-action pairs $(\mathbf{s}_t, a_t)$ can be transferred to $(\widetilde{b}_t, a_t)$,



(a) (b)

Figure 6.6: Two different situations where the agent should follow the same turning angle, $\tilde{b} = 270°$.

and their values are updated, similar to the SMDP Q-learning method in Eq. (6.7):

$$Q_{L3,T1}(\tilde{b}_t, a_t) = (1-\delta)Q_{L3,T1}(\tilde{b}_t, a_t) + \delta\left[\gamma^{\tau-1-t}\cdot r_\tau + \gamma^{\tau-t}\cdot \max_{a_\tau \in \mathscr{A}_{\tilde{b}_\tau}} Q_{L3,T1}(\tilde{b}_\tau, a_\tau)\right]. \tag{6.12}$$

Level 3 sub-task 2

The second sub-task, L3.T2, is learning to prevent collisions online with the Q-learning method. After taking an action, the agent may receive an immediate reward r_{t+1}, which can not only be a collision penalty (external reward) but can also be a getting-close penalty (internal reward) if the agent observes an obstacle in front. This task does not directly relate to the agent's heading, which can, therefore, reduce the dimension of the value function to $\mathscr{S}_f \times \mathscr{S}_l \times \mathscr{S}_r \times \mathscr{A}$. The value function is updated as follows:

$$Q_{L3,T2}(\mathbf{s}_t^{T2}, a_t) = (1-\delta)Q_{L3,T2}(\mathbf{s}_t^{T2}, a_t) + \delta\left[r_{t+1} + \gamma \max_{a_{t+1} \in \mathscr{A}_{\mathbf{s}_{t+1}^{T2}}} Q_t(\mathbf{s}_{t+1}^{T2}, a_{t+1})\right], \tag{6.13}$$

where $\mathbf{s}^{T2} = [s_f, s_l, s_r]$.

The lower levels (L2 and L3) also keep exploring within their state-action spaces with an ϵ-greedy strategy, which chooses the greedy policy with probability $1-\epsilon$, and a random action with probability ϵ. The exploration rate in this chapter is initialized as $\epsilon_0 = 0.5$ and is decayed by 0.95 after each update until it reaches the minimum exploration rate $\epsilon = 0.1$.

6.5. Results and Discussion

In this section, three simulation experiments are carried out for online guidance and navigation problems to demonstrate different capabilities of hHRL. First, the agent learns to prevent collisions online and to episodically improve the performance of approaching the target in *maze A*. The result will be compared to those using a 'flat' Q-learning and a hierarchical Q-learning algorithm. Second, the hHRL method is applied to a different, expanded maze to validate the transferability of learning. Next, this method is also applied to a non-stationary environment using modified sensors and a partial map.

6.5.1. LEARNING EFFICIENCY IN AN A PRIORI UNKNOWN MAZE A

The proposed hHRL method is applied to the guidance and navigation problem in *maze A* as Fig. 6.3. The maze is a priori unknown to the agent. In addition, the initial position in each episode is also unknown and can vary. These simulations use pseudo-random initial positions (pre-determined and in the northeast area in *maze A*) for 30 episodes, where the first and final episodes start from the most northeast position, for a fair comparison with the 'flat' Q-learning (fQ) and the hierarchical Q-learning (hQ) algorithm [131].

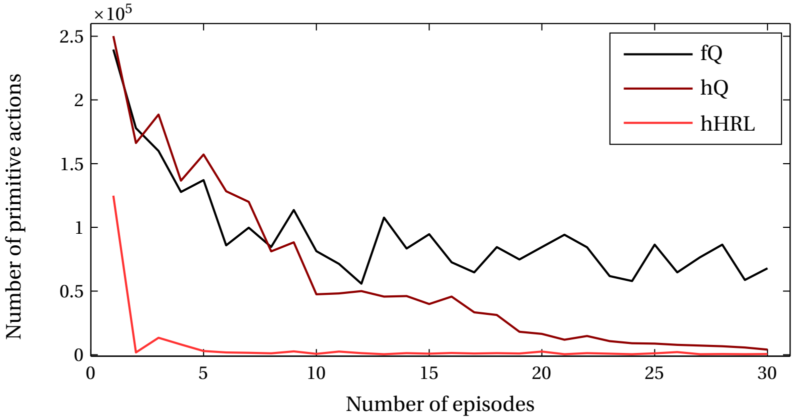


Figure 6.7: The number of primitive actions taken in each episode with a 'flat' Q-learning (fQ) [131], a hierarchical Q-learning (hQ) [131], and the proposed hHRL algorithm (averaged of 5 runs).

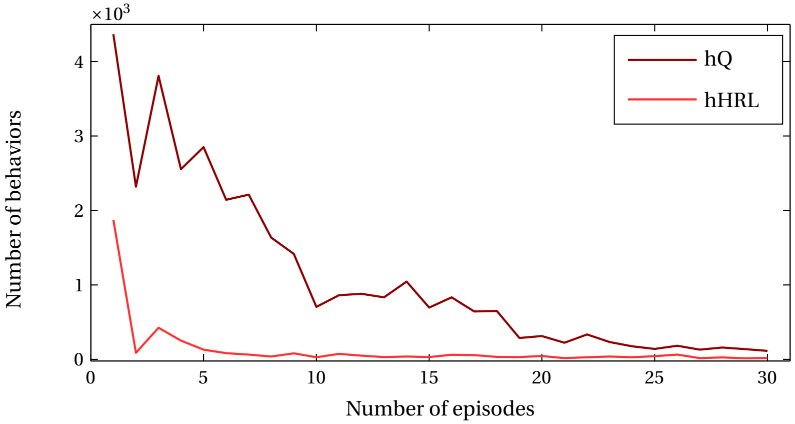


Figure 6.8: The number of macro actions taken in each iteration with a hierarchical Q-learning (hQ) [131] and the proposed hHRL algorithm (averaged of 5 runs).

Figure 6.7 compares the numbers of primitive actions taken in each iteration using the fQ, hQ algorithms from [131], which has the same agent, same maze, and same environment decomposition as this chapter, and the proposed hHRL method. With fQ, the agent learns how to prevent collisions in the first several episodes. However, the performance will not be improved significantly hereafter. There are two main reasons: first, the huge number of actions taken before reaching the target area makes the effect of the terminal reward propagate very slowly to the experienced states; and second, the partial observability leads to the ambiguity in real absolute states, which prevents the convergence of observed state-action values. These features impair the agent's ability to perform the second task: approaching the target area.

On the other hand, the HRL algorithms, including hQ and hHRL, improve the performance of both the collision avoidance objective and the searching for the target area objective. The low-level agent focuses on avoiding obstacles and following high-level instructions. The high-level agent focuses on the exploration of the state space and finds the optimal behaviors to get to the target area as soon as possible. In Fig. 6.8, there is a clear trend of decreasing number of the behaviors during learning. This result indicates that the agent improves the performance also from a higher level.

Compared to hQ, the proposed hHRL algorithm improves the performance considerably and much faster even from the first episode, as depicted in Figs. 6.7 and 6.8. There are several reasons:

1) The hHRL algorithm separates different objectives into different sub-tasks and efficiently uses different value functions for each sub-task, e.g., Eq. (6.11) for L3.T1 and Eq. (6.12) for L3.T2. The adaptations of these value functions are independent from each other.

2) The hHRL method allows for different, appropriate methods for different sub-tasks to effectively tackle each sub-task. The MC off-line learning in L2.T2 is more efficient in the use of the final reward than the Q-learning in hQ.

3) The hHRL method uses a better strategy to connect sub-tasks in the same level as in Eq. (6.9).

4) The hHRL algorithm is open to being expanded and has a decision level to trade between the exploration and exploitation in macro-state levels, which improves the learning efficiency in a totally unknown environment, e.g., during the first episode.

Figure 6.9 shows the agent's internal memory during online learning in its 30th episode, starting from the most northeast position. This episode takes 272 primitive actions, 15 behaviors, and experiences 12 macro states. The greedy behaviors in the learned external map after 30 episodes are presented in Fig. 6.10. Since each macro state consists of an area of fixed size, the corridor area is not completely separated from the inaccessible area with walls. Greedy behaviors are also assigned to those macro states with partly corridor, partly inaccessible area with walls, and partly inexperienced environment. The agent creates the map in Fig. 6.10 by stitching the constructed maps from the internal memory after each episode. After 30 episodes (actually from around the 15th episode), the agent, using the hHRL algorithm, has the ability to find a near-optimal path online in the micro-state level, and optimal behaviors in the macro-state level.

Note that the hHRL method is not very sensitive to the pre-determined parameters: macro-state size in the state space decomposition, different reward assignment systems,

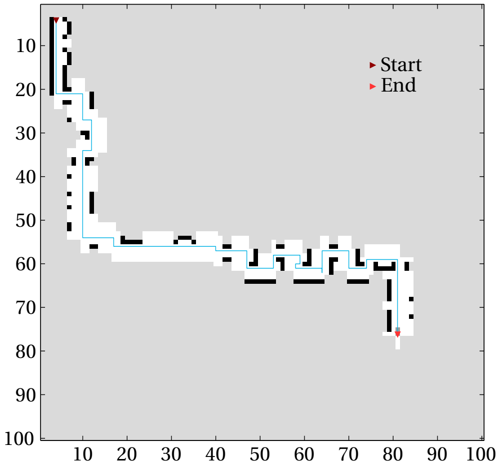


Figure 6.9: An example of agent's internal memory in the 30th episode, which takes 272 primitive actions, 15 behaviors, and experiences 12 macro states. The blue line shows the trace of the agent in this episode. The white and black areas are experienced or observed states and obstacles. The gray color indicates that those areas haven't been experienced or observed in this episode.

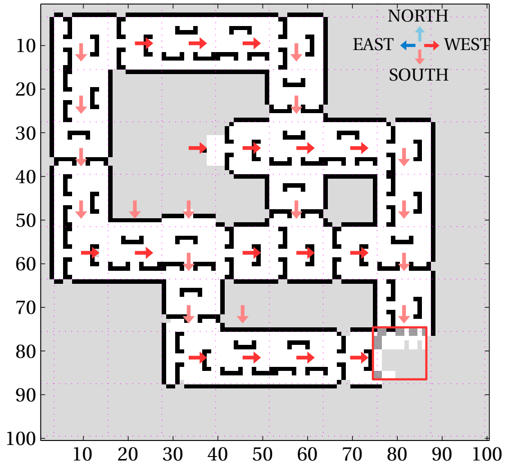


Figure 6.10: The greedy behaviors in the external map after 30 iterations with hHRL in *maze A*. The grid with pink dotted line decomposes the environment into macro states. The light gray color indicates that those areas haven't been experienced or observed. The dark gray areas are experienced target states and are clustered with a red rectangle.

the forgetting factor, and the learning-rate. Although in the shown results, a macro state encompasses 12 × 12 micro states, which is the same as in [131], simulations with other macro-state sizes, e.g., 8 × 8 or 6 × 6, can also reach a similar level of performance. The rewards used in different levels are independent and will not affect the learning results. However, the rewards assigned at the same level or for the same sub-task may change the learning and the performance. The above results use a reward assignment system adhering to ***Rule 5***. When the rewards change reasonably, e.g. if the immediate penalty for obstacles in front of the agent '-2', '-1', '-0.5' in section 6.4.2 is changed to '-1', '-0.8', '-0.6', the learning performance will not change. This chapter uses a same forgetting factor and learning-rate, as $\gamma = 0.2$ and $\delta = 0.8$, for all sub-tasks. These parameters are also changeable and will not significantly affect the learning performance within a proper range.

6.5.2. Transferability of Learning to a New Maze B

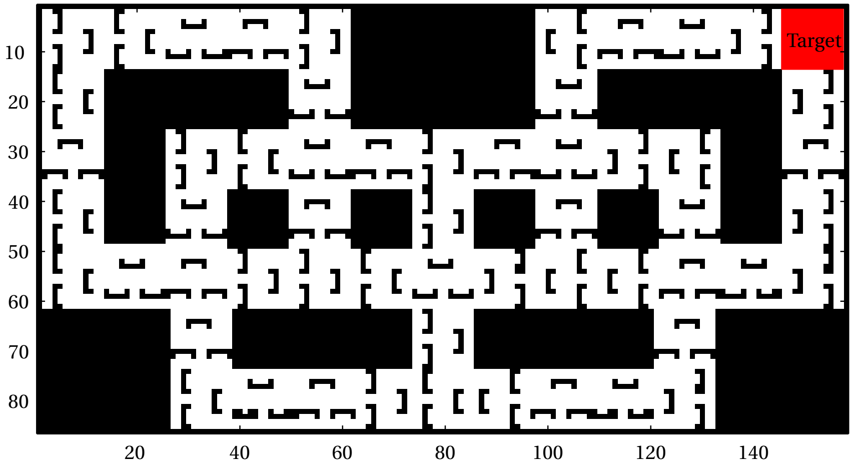


Figure 6.11: A guidance and navigation problem with obstacles and a target area in *maze B*.

The hHRL method allows for the transfer of learned results across tasks, such as guidance and navigation in a new and unknown environment. Figure 6.11 shows another maze expanded from *maze A*. This *maze B* has about twice the number of absolute states as *maze A* and a different target area in the northwest of the new maze. The agent, which has been trained in *maze A*, can still use the low-level results in L3.T1 and L3.T2 and only need to learn a new higher level policy.

Figure 6.12 presents the agent's internal memory during an online learning in its 30th episode in *maze B*. This episode takes 520 primitive actions, 23 behaviors, and experiences 20 macro states. The agent creates a map of *maze B* in its external memory with greedy behaviors, as depicted in Fig. 6.13. Although the starting positions are all in the northeast area in *maze B* (pseudo-random, same as *maze A*), most of the greedy behaviors in Fig. 6.13 are optimal. With more learning episodes and starting from different locations, the agent will explore all possible actions and exploit those experience to learn

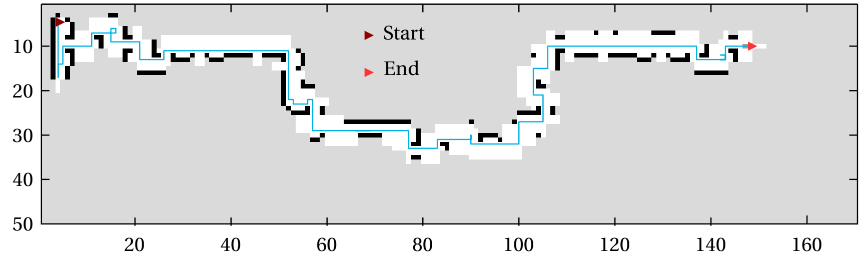


Figure 6.12: An example of agent's internal memory in the 30th episode, which takes 520 primitive actions, 23 behaviors, and experiences 20 macro states. The blue line shows the trace of the agent in this episode. The white and black areas are experienced or observed states and obstacles. The gray color indicates that those areas haven't been experienced or observed in this episode.

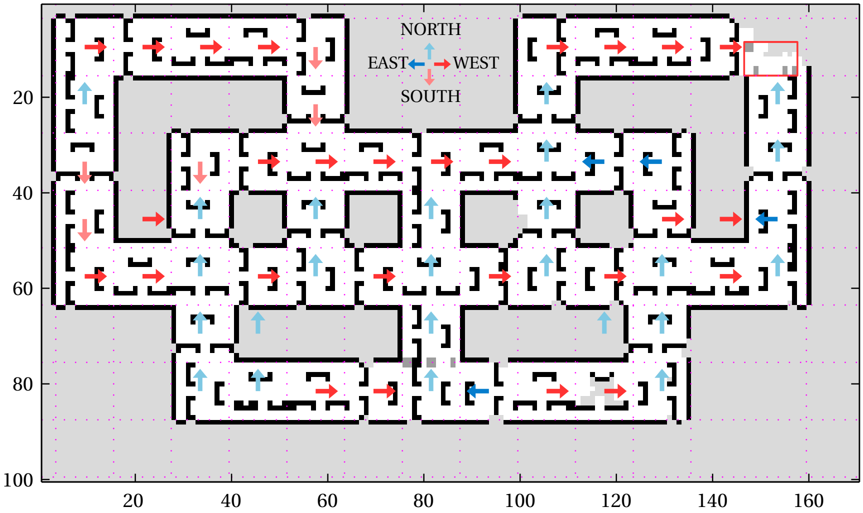


Figure 6.13: The greedy behaviors in the external map after 30 episodes with hHRL in *maze B*. The grid with pink dotted line decomposes the environment into macro states. The light gray color indicates that those areas haven't been experienced or observed. The dark gray areas are experienced target states and are clustered with a red rectangle.

to accomplish the tasks better. The agent will, therefore, has the ability to find the hierarchical optimal path online from any initial position.

The previous experience in *maze A* and the learned results can be transferred to similar tasks in *maze B*. Figure 6.14 compares the learning performance in *maze B* with and without pre-training in *maze A*. The agent, pre-trained in *maze A*, takes less primitive actions and behaviors since the first episode, and its policy converges since around the 10th to 15th episode, as depicted in Figs. 6.14(a) and (b). On the other hand, the agent, without pre-training in *maze A*, takes more primitive actions and behaviors and reaches the same level of performance as the pre-trained one since around the 20th to 25th episode. In addition, the pre-trained agent can effectively prevent collisions in a new environment, as presented in Fig. 6.14(c), which is useful particularly in practice. The results show that the proposed hHRL method owns an ability to transfer the learned results across tasks and can speed up learning for different, more complex tasks or environments.

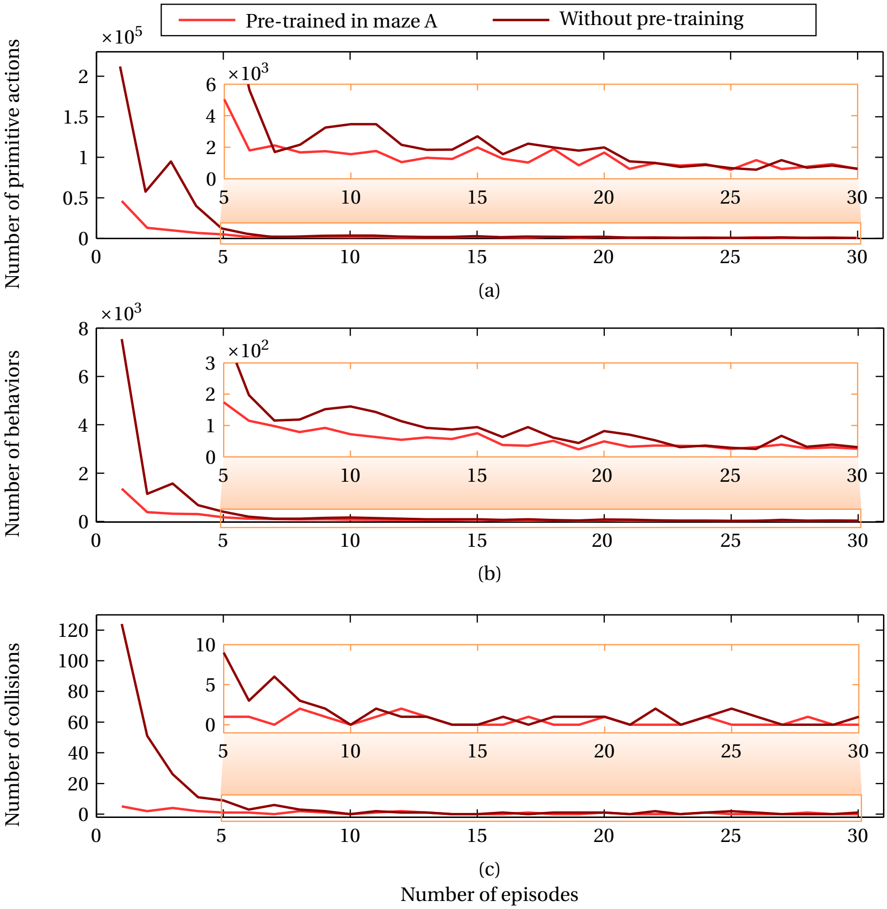


Figure 6.14: Numbers of (a) primitive actions, (b) behaviors, and (c) collisions during the first 30 episodes in *maze B* using the proposed hHRL algorithm, with and without pre-training in *maze A* (averaged of 5 runs).

6.5.3. APPLICABILITY IN NON-STATIONARY ENVIRONMENTS

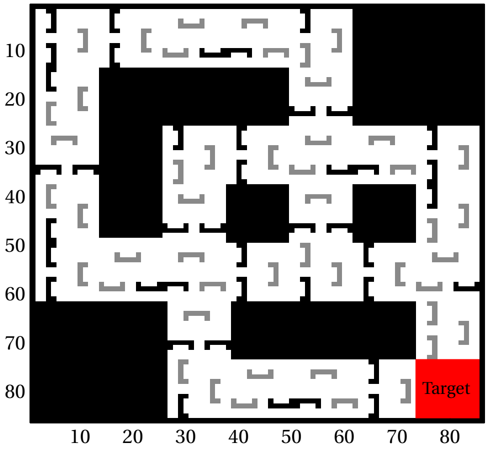


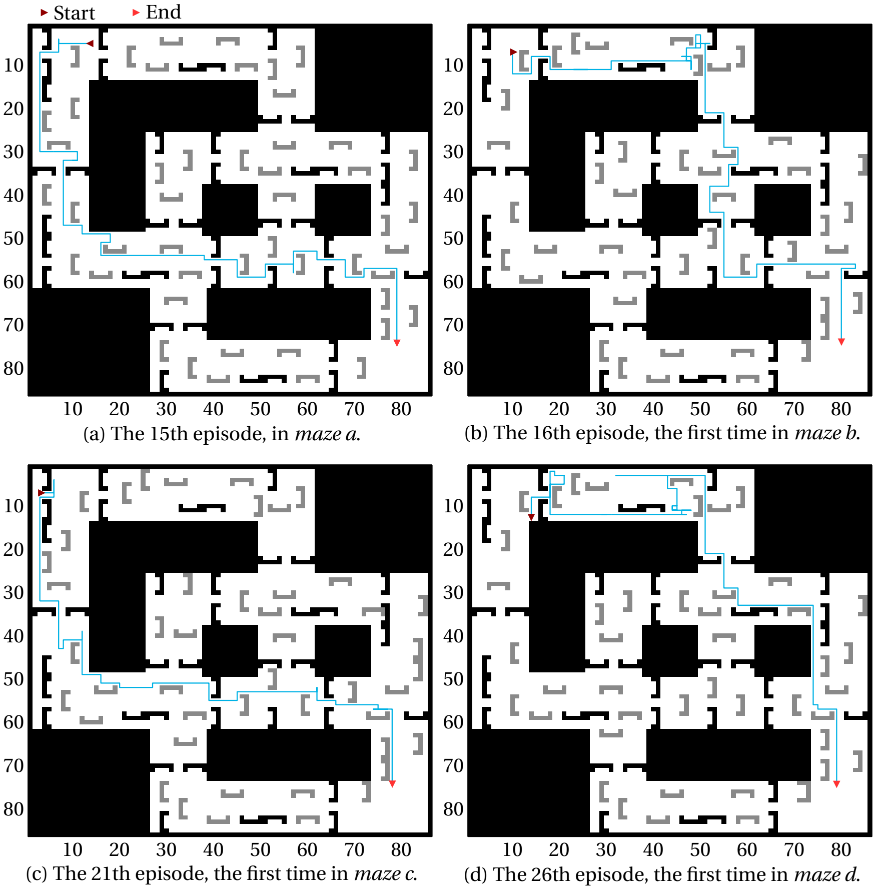
Figure 6.15: A guidance and navigation problem in non-stationary environments. The black color represents stationary obstacles, while the gray color represents non-stationary obstacles.

The proposed hHRL method can also be used for non-stationary environments, as depicted in Fig. 6.15, only by changing the complete map in external memory to a partial map. This partial map only contains the static environment, for macro-state level localization. The non-stationary surroundings are treated as temporary obstacles and will be forgotten when the current episode ends. The sensors equipped on the agent are modified to be able to measure if the obstacles are stationary or non-stationary, as represented by the black and gray colors in Fig. 6.15.

Instead of changing any of the value functions in section 6.4.4, the agent pass on different sensor information to the sub-tasks in different levels. The micro-state level, L3, only concerns whether there are observed obstacles instead of what kind of obstacles are there. On the other hand, the macro-state level, L2, treats non-stationary obstacles as empty areas. The internal and external memory will store partial maps, which only contain stationary obstacles.

This section focuses on the validation of the proposed method in non-stationary environments, especially on the consistency of online performance in never experienced environments. The agent starts with the original maze in Fig. 6.15, denoted as *maze a*, for 15 episodes, and from the 16th episode onwards, the non-stationary obstacles may change their position every per 5 episodes. Figure 6.16 (a) shows the agent's trace in the 15th episode in *maze a*. The environment changes to *maze b*, *maze c*, and *maze d* in the 16th, 21th, and 26th episodes for the first time, where the traces of the agent are recorded in Figs. 6.16 (b), (c), and(d), respectively.

Figure 6.17 presents the greedy behaviors in the learned partial map after 30 episodes in the external memory. In addition, the number of primitive actions and behaviors taken in each episode are illustrated in Fig. 6.18. The policy converges since around the 10th to 15th episode. From the 16th episode onwards, the averaged number of primitive actions and behaviors are 686 and 33. And the averaged numbers over the episodes in



(a) The 15th episode, in *maze a*.

(b) The 16th episode, the first time in *maze b*.

(c) The 21th episode, the first time in *maze c*.

(d) The 26th episode, the first time in *maze d*.

Figure 6.16: The traces of the agent in non-stationary environments.

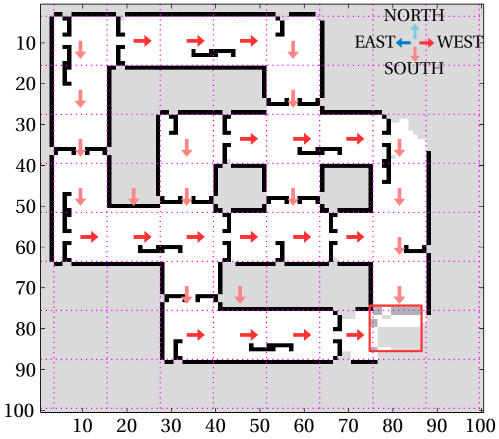


Figure 6.17: The greedy behaviors in the external (partial) map after 30 episodes with hHRL in the non-stationary environment. The black color represents stationary obstacles. The white color indicates that those areas are empty spaces or non-stationary obstacles.

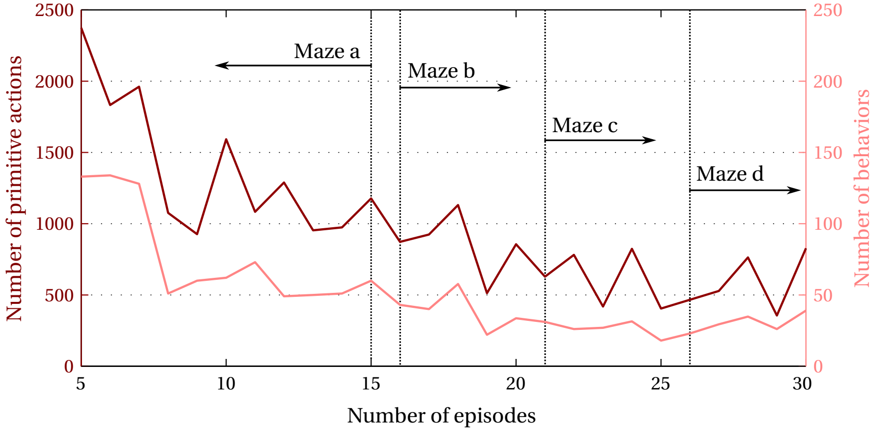


Figure 6.18: Numbers of primitive actions and behaviors during the first 30 episodes in non-stationary environments, using the proposed hHRL algorithm and a partial map (averaged of 5 runs).

new environments (16th, 21th, and 26th episodes) are 655 and 32, which are in the same level of performance as the average, and are even slightly better. The above results indicate that the proposed algorithm is applicable to non-stationary environments without the loss of efficiency.

6.6. Conclusion

This chapter proposes a systematic hybrid Hierarchical Reinforcement Learning (hHRL) method for multiple-objective problems with partial observability. This method uses different methods, types of state information, and reward assignment systems to optimize the learning. This chapter formulates the explicit rules of establishing the hierarchies, decomposing the tasks, and assigning the rewards. The detailed implementations of the proposed hHRL method are presented for an online guidance and navigation task.

The proposed method is applied to a benchmark maze, Parr's maze, with partial observability to prevent collision online and to episodically improve the performance of approaching the target. The result is compared to a 'flat' Q-learning and a hierarchical Q-learning method and indicates that the proposed hHRL method is more efficient in dealing with the 'curse of dimensionality' and in reducing the uncertainty or ambiguity in a higher level. The learned results are, then, applied to a different, expanded maze, which validates that learned results can be transferred across tasks to speed up learning for different environments. In addition, the same method is also applied to non-stationary environments with modified sensors and a partial map. The hHRL method, using relative micro states and absolute macro states at different levels, allows for learning in non-stationary environments without loss of efficiency.

This chapter focuses on the partial observability, the transfer of learning, and the applicability in non-stationary environments in an online guidance and navigation task. Future research is recommended to be undertaken in the following areas: 1) The hHRL method can be applied to more realistic environments in continuous (both relative and absolute) state spaces. 2) Approximate dynamic programming methods, presented in Chapters 2 to 5, can be used in the lower level to combine the control of nonlinear, unknown aerospace systems to this online guidance and navigation method. 3) The mapping of and adaptation to changes in non-stationary environments, with limited sensing information, and changing over time instead of over episodes, are worth to be investigated. The proposed hHRL is open to being expanded both upwards to tackle more complex, multiple-objective tasks and downwards to control more complex, nonlinear, or continuous systems.

7

Conclusions and Recommendations

This chapter summarizes the research findings, by first discussing how the proposed methods address the research questions posed in Chapter 1. Then, the main contributions are presented to draw final conclusions of the entire dissertation. Lastly, several suggestions and recommendations are proposed for future work.

7.1. Discussion

This dissertation started with three challenges of applying current Reinforcement Learning (RL) controllers to aerospace systems without knowing the model of the system or the environment. The challenge of efficiency is posed to the control of unknown, nonlinear aerospace systems, especially when the system is time-varying and partially observable. When applied to more general nonlinear systems and tasks, the RL controller must adapt in real-time also for systems without representative simulation models and for faulty systems. The high-level guidance and navigation problems, with multiple objectives, face the challenge of designing systematic and transferable RL methods with explicit rules. These three challenges are reflected in the main research question:

Main Research Question

How can aerospace systems exploit RL methods to improve the autonomy and online learning with respect to the a priori unknown system and environment, dynamical uncertainties, and partial observability?

This main research question is addressed in three parts, for three specific RL methods and applications: Approximate Dynamic Programming (ADP) for control problems with an approximately convex true cost-to-go, Adaptive Critic Designs (ACDs) for general nonlinear control problems, and Hierarchical Reinforcement Learning (HRL) for high-level guidance and navigation.

7.1.1. Incremental Approximate Dynamic Programming

Linear Approximate Dynamic Programming (LADP) methods have successfully combined RL methods and optimal control for unknown linear time-invariant systems. These methods allow for numerically approximating the convex true cost-to-go without knowing the system model, and keep the control algorithm efficient and mathematically explicit. However, most aerospace systems are nonlinear and may involve dynamical uncertainties and partial observability, and cannot use LADP methods. **Part I** of this dissertation, therefore, aimed at answering the following research question:

> RQ1: How to generalize LADP to deal with nonlinear and/or time-varying systems, model mismatch, and partial observations, while retaining the efficiency and mathematical explicitness?

This research question is answered through the development of *incremental* Approximate Dynamic Programming (iADP) methods. Instead of using nonlinear approximators to approximate the true cost-to-go, iADP methods use an (extended) incremental model to deal with the nonlinearity of unknown systems and uncertainties of the environment. These methods can still apply a quadratic cost function to generate an efficient and mathematically explicit optimal control algorithm. These methods do not need any a priori knowledge of the system dynamics, online identification of the global model, nor even an assumption of the time scale separation, but only an online identified (extended) incremental model.

Chapter 2 introduces the iADP method to solve regulation problems for nonlinear systems. When the direct measurement of the full state is available, the incremental model can be identified to predict the next state. With this prediction, the optimal control increment can be calculated with the iADP algorithm based on Full State feedback (iADP-FS). When the only measurements are the input and the output of the dynamical system, the iADP algorithm based on OutPut feedback (iADP-OP) is developed, to optimize the control increment by reconstructing an extended incremental model. Both algorithms are applied to an unknown missile model, with nonlinear aerodynamic uncertainties, to optimize the flight controller iteratively. The presented results demonstrate that the proposed method improves the closed-loop performance of the nonlinear system, while keeping the design process simple and systematic.

The concept of iADP is further expanded in **Chapter 3** to tracking problems for Multiple-Input Multiple-Output (MIMO) nonlinear systems and to partial observable control problems. Because iADP methods have a separate structure to represent the local system dynamics, the cost function can be less dependent on the system or the reference, and only needs to be a rough approximation of the cost-to-go. This approximation is a quadratic function only of the current tracking error, without expanding the dimension of the state space for the cost function to an augmented one.

Two observability conditions are considered in this tracking control problem. When the direct measurement of the full state is available, the incremental model can be online identified to design the optimal control increment. In addition, when the only measurement is the output tracking error, which is the result of tracking a stochastic dynamical reference, the system becomes partially observable. The observations are used to identify the extended incremental model and to predict the next output tracking error for the optimal tracking control. For each observability condition, an off-line learning algorithm is applied to improve the policy iteratively until it is accurate enough, and hereafter an online algorithm is applied to update the policy recursively at each time step. The recursive algorithms can also be used online in real systems, which may be different from the system model used in the iterative learning stage. These algorithms are applied to an attitude control problem of a simulated satellite disturbed by liquid sloshing. The results demonstrate that the proposed algorithms accurately and adaptively deal with time-varying internal dynamics while retaining efficient control, especially for unknown nonlinear systems with only partial observability.

Part I of this dissertation illustrates two aerospace control problems where the system does not have enough information to infer its full state: 1) the regulation of deterministic systems, which belongs to output feedback methods, and 2) the tracking problems involving stochastic, time-varying dynamics in reference signals, which are referred to as POMDPs. In these cases, both the system output in **Chapter 2** and the output tracking error in **Chapter 3** are mathematically proven to be predictable with extended incremental models, instead of reconstructing the full model or estimating the full state. The necessary conditions of extended incremental models are presented for generating the optimal control increment efficiently and explicitly.

7.1.2. ONLINE ADAPTIVE CRITIC DESIGNS BASED ON THE INCREMENTAL MODEL

Although the proposed methods in Part I are beneficial for most tracking control problems with approximately convex cost-to-go, the quadratic function may not be suitable for more general nonlinear systems and tasks. ACDs can be used to control nonlinear systems by exploiting nonlinear function approximators. In ACDs, an accurate global system model still plays an important role. This model is, however, identified off-line using representative simulation models, which may be difficult to obtain and are often not accurate themselves. In addition, the online adaptation of the global system model also needs to be sufficiently quick and smooth, to deal with unforeseen dynamics in the system, uncertainties in the environment, and unexpected changes due to failures. Therefore, the research question in **Part II** is formulated as follows:

RQ2: How to devise online ACDs and improve the online adaptability, to cope with internal uncertainties, external disturbances, and even sudden faults?

This research question is answered through the development of online ACDs based on the *incremental model*. ACDs can generally be categorized into three groups: 1) Heuristic Dynamic Programming (HDP), 2) Dual Heuristic Programming (DHP), and 3) Globalized Dual Heuristic Programming (GDHP). Besides, AD variations of these three original versions have been developed by directly connecting the output of the actor to the input of the critic. The AD forms may reduce the dependency on the system model. However, from a theoretical point of view, the actor output is not necessarily an input to the critic; and from a practical perspective, the extra input will increase the complexity of the critic. Furthermore, previous studies comparing ACDs and their AD forms have reported that ACDs have higher success rates and online adaptability [51, 54]. Therefore, this dissertation focuses on action independent ACDs, specifically HDP and DHP.

Chapter 4 proposes an online HDP method based on an **incremental model**, called IHDP, to design adaptive controllers without a priori knowledge of the system dynamics. This method replaces the global system model approximator with an incremental model. This approach, therefore, does not need off-line training stage and may accelerate online learning. The IHDP method is compared with conventional HDP in an online tracking control of the unknown nonlinear missile model. The results show that the presented IHDP method speeds up the online learning, has a higher precision, and can deal with a wider range of initial states than the conventional HDP method. In addition, the IHDP method is also applied to the MIMO satellite attitude tracking control disturbed by liquid sloshing and with sudden external disturbances. The simulation results also demonstrate that the IHDP method is adaptive and robust to internal uncertainties and external disturbances.

Chapter 5 further combines the advantages of DHP and *incremental model* to develop an online DHP method, called IDHP. The IDHP method uses a Recursive Least Square (RLS) approach to identify in real-time the incremental model instead of the global system model. In addition to the online reference tracking problem, a Fault-Tolerant Control (FTC) task is performed using IDHP and conventional DHP, in the presence of sudden changes of the original system dynamics, which could be caused by sudden faults. The results demonstrate that the IDHP method can successfully control a new and unstable system adaptively before the states diverge, where DHP fails. To further validate the robustness of the proposed IDHP method, high-frequency measurement noise is superimposed to the measurements of system states. The simulation results indicate that the IDHP method is not sensitive to the measurement noise.

To accelerate the online learning is of great practical value, especially when a priori knowledge is unknown, the system is initially unstable, or the system or environment changes suddenly. **Part II** proposes to use an incremental model to replace the global system model in conventional ACDs. This linear model, identified from a series of most recent measurements, provides more accurate local system dynamics and state prediction with less computation. In addition, the adaptation of the critic and/or the actor is also simplified with directly identified local derivatives of the next state. The proposed online ACD methods relieve the off-line learning phases and are validated to be online efficient with internal uncertainties, external disturbances, and even sudden faults.

7.1.3. HYBRID HIERARCHICAL REINFORCEMENT LEARNING FOR HIGH-LEVEL GUIDANCE AND NAVIGATION

High-level guidance and navigation problems often have high-dimensional spaces, multiple objectives, and consequently a large number of states and actions, which is known as the 'curse of dimensionality'. In addition, aerospace systems often have partial observability instead of a perfect perception of their environment. Recent research has sought to deal with these problems by using HRL. However, the explicit rules of establishing the hierarchies often need expert knowledge, and the learned results in one application cannot be directly used in other applications. **Part III** of this dissertation, therefore, aimed at answering the following research question:

> RQ3: How to establish a systematic HRL controller that deals with multiple objectives and partial observability, possesses transfer learning ability, and utilizes diverse RL methods?

To answer this question, **Chapter 6** proposes a systematic *hybrid* Hierarchical Reinforcement Learning (hHRL) method for guidance and navigation problems with multiple objectives and partial observability. This method consists of several hierarchical levels, where each level uses different methods to optimize the learning with different types of information and objectives. The chapter formulates the explicit rules of establishing the hierarchies, decomposing the tasks, and assigning the rewards. Detailed implementations of the proposed hHRL method are presented for an online, multi-objective guidance and navigation task: approaching a target area while avoiding obstacles.

The proposed method is first applied to a benchmark maze, to prevent collision online and to episodically improve the performance of approaching the target. The result is compared to a 'flat' Q-learning and a hierarchical Q-learning method [131] and indicates that the proposed hHRL method is more efficient in dealing with the 'curse of dimensionality' and in reducing the uncertainty or ambiguity in a higher level. The learned results are then applied to a different, expanded maze, which validates that learning results can be transferred across tasks to speed up learning in new tasks or environments. Lastly, the same method is applied to a non-stationary environment with modified sensors and a partial map. The hHRL method, using relative micro states and absolute macro states in different hierarchical levels, allows for learning in non-stationary environments without loss of efficiency. These results indicate that the proposed hHRL method can help to accelerate learning, to alleviate the 'curse of dimensionality' in complex decision-making tasks, to naturally reduce the uncertainty or ambiguity at higher levels, to transfer the learned results within and across tasks efficiently, and to be applied to non-stationary environments. This proposed method can potentially yield a near-optimal policy hierarchically for autonomous guidance and navigation, without a priori knowledge of the environment.

7.2. FINAL CONCLUSIONS

In this dissertation, several methods are proposed to improve the intelligence and autonomy of aerospace systems, mainly from the following three perspectives:

1. *Enhance the adaptability and efficiency of low-level control*

   The low-level control ability is the foundation of high-level guidance and navigation, and limits the performance of the whole autonomous control system, especially for aerospace systems. The unknown nonlinear dynamics, internal uncertainties, and unpredictable changes are all in need of adaptive and efficient control methods. This dissertation proposes to use incremental techniques in several conventional RL methods to improve their adaptability and efficiency. These methods maintain the mathematical explicitness and/or generalization of conventional RL methods that are applied to a priori unknown systems, and take advantage of an incremental model to cope with system nonlinearities, uncertainties, and even system faults.

This dissertation uses an extended incremental model to deal with partial observability of continuous systems. This extended incremental model is identified online from the partial observations, and can be directly used to generate an optimal control. Compared to estimating the full state or reconstructing the full model, the proposed method is more efficient and can reduce computation time delays.

2. *Improve the intelligence and online learning ability of guidance, navigation, and control*

   Reinforcement learning is inspired by behaviors of human beings and animals. They learn from the rewards, penalties, and even failures, and these processes are always 'online'. Similarly, the online learning ability is of paramount importance in intelligent control systems, and particularly RL controllers. In this dissertation, the proposed iterative learning algorithms are called off-line to be distinguished from the step-by-step online learning. Nevertheless, from an RL perspective, they are also online learning because they start learning without a priori information of the system and the environment. The proposed methods and algorithms in this dissertation fulfill different degrees of online learning ability requirements.

   IADP belongs to critic-only methods, which only have state value functions and rely on the optimality principle to generate an action. The control policy is closely dependent on the value function. These methods, therefore, keep the policy unchanged during each iteration, and evaluate and improve the policy iteratively until converged. The converged policy can be used as an initial policy for mismatched systems and similar control tasks with further online, recursive adaptation. Online ACDs, on the other hand, are actor-critic methods, which separate the policy evaluation and improvement into different memory structures. Although these two approximations update alternatively based on each other, their adaptation is not necessarily synchronized. In other words, the critic adaptation usually happens earlier than the actor. This time difference also provides a chance for the critic to evaluate a relatively consistent policy and makes it possible to prevent the iterative learning stage. For guidance and navigation problems, the hHRL method involves both iterative and step-by-step online adaptation for different value functions, according to the desired behaviors and reward assignments. The system receives penalties after each collision, with which the related value functions can be updated immediately. For approaching the target, the system only receives a final reward after reaching this goal, and then updates the related value functions iteratively.

3. *Create a well-organized hierarchy to ensure coordination between each level*

The last part of this dissertation proposes a hHRL method consisting of several levels, where each level uses different methods to optimize the learning with various types of information and objectives. The explicit rules of establishing the hierarchies are presented, based on the state abstraction levels. These rules have a clear and well-organized hierarchy and ensure the coordination between each level by properly assimilating independent, diverse learning processes. In addition, the proposed hHRL is open to being expanded both upwards, to tackle more complex, multiple-objective tasks, and downwards, to control more complex, nonlinear, or continuous systems, and allows for human involvement, such as setting reference values or rules, at all levels.

7.3. RECOMMENDATIONS

The proposed methods and results in this dissertation provide the following insights for further research:

- This dissertation focused on the method development, theoretical analysis, and simulation experiments. Further research should, therefore, concentrate on the validation of the proposed methods on more complex and higher degree-of-freedom aerospace models and then on real systems. Therefore, practical issues, such as reducing noise and time delay in the measurement, using elaborate fault detection and isolation methods, and improving the accuracy and real-time capability of the incremental model identification methods, need to be taken into account.

- In aerospace applications, partial observability often occurs when the system does not have enough information to infer all of its states. This dissertation proposes several algorithms for partially observable conditions by directly using relative states. These methods are worth to be further investigated on more realistic aerospace applications, such as spacecraft rendezvous/docking problems, missile guidance, UAV swarm flight control, and aircraft formation flying.

- The proposed iADP method uses a quadratic cost function for control problems with an approximately convex true cost-to-go. This type of state value function is the essential element of generating an mathematically explicit optimal control solution. When applied to more complex nonlinear systems and tasks, the use of general nonlinear function approximators, such as artificial neural networks and radial basis functions, will end up with an implicit solution. Therefore, to generalize iADP to more complex control problems, research on using mathematically explicit, nonlinear approximators, such as piecewise quadratic value functions and multivariate splines [152], is highly recommended.

- The proposed online ACD methods can be used not only in low-level control tasks, as presented in this dissertation, but also in high-level guidance and navigation tasks with highly nonlinear value functions. Further experimental investigations into different tasks are recommended to validate the generalization of online ACDs.

- This dissertation investigated the online HDP and DHP methods based on an incremental model. Another ACD method, GDHP, combines the advantages of HDP and DHP, from a theoretical point of view. However, its computational complexity is much higher than the other two types of ACDs because it needs to calculate second derivative terms. It would be interesting to use an incremental model in GDHP to reduce the computational burden.

- The incremental model is based on the first order Taylor series, hence involves a linear approximation around the current system state. This linear model works well for the examples included in this dissertation, but may work less well for systems with high nonlinearities or fast-changing control inputs. A future investigation into Volterra series [153, 154] as a possible nonlinear incremental model, would be useful, especially for time-delayed measurements, low-frequency measurements, and internal/actuator dynamics.

- Learning in non-stationary environments attracts increasing attention in RL and aerospace systems due to its importance for practical applications. This dissertation applies hHRL to learn in a non-stationary environment with modified sensors and a partial map. However, the sensors are not always able to measure if an obstacle is stationary or non-stationary. Therefore, future research will need to use limited sensor information to distinguish non-stationary and stationary obstacles, such as by using a probability map instead of a fixed (partial) map. In addition, the non-stationary environments changing over time instead of over episodes are also worth to be investigated.

- The hHRL method is a systematic, multiple-level method. It is applied to a simplified discrete system in a discrete guidance and navigation environment in this dissertation. Further research is recommended in more realistic environments and aerospace systems in continuous state spaces, by expanding downwards with a control level or by directly replacing the discrete micro-state level RL method with the ADP methods proposed in this dissertation.

References

[1] P. J. Antsaklis, K. M.Passino, and S. J. Wang, *An introduction to autonomous control systems,* IEEE Control Systems **11**, 5 (1991).

[2] D. D. Woods, *The risks of autonomy: Doyleś catch,* Journal of Cognitive Engineering and Decision Making **10**, 131 (2016).

[3] M. Keennon, K. Klingebiel, H. Won, and A. Andriukov, *Development of the nano hummingbird: A tailless flapping wing micro air vehicle,* in *AIAA aerospace sciences meeting* (AIAA Reston, VA, 2012) pp. 1–24.

[4] R. J. Wood, *The first takeoff of a biologically inspired at-scale robotic insect,* IEEE transactions on robotics **24**, 341 (2008).

[5] B. M. Finio, J. K. Shang, and R. J. Wood, *Body torque modulation for a microrobotic fly,* in *2009 IEEE International Conference on Robotics and Automation, ICRA'09.* (IEEE, 2009) pp. 3449–3456.

[6] G. C. H. E. de Croon, K. M. E. De Clercq, R. Ruijsink, B. Remes, and C. De Wagter, *Design, aerodynamics, and vision-based control of the delfly,* International Journal of Micro Air Vehicles **1**, 71 (2009).

[7] G. De Croon, M. Perçin, B. D. Remes, R. Ruijsink, and C. De Wagter, *The DelFly: design, aerodynamics, and artificial intelligence of a flapping wing robot* (Springer, 2015).

[8] T. Lombaerts, E. V. Oort, Q. P. Chu, J. A. Mulder, and D. Joosten, *Online aerodynamic model structure selection and parameter estimation for fault tolerant control,* Journal of Guidance, Control, and Dynamics **33**, 707 (2010).

[9] M. Sghairi, A. De Bonneval, Y. Crouzet, J. Aubert, and P. Brot, *Challenges in building fault-tolerant flight control system for a civil aircraft,* IAENG International Journal of Computer Science **35**, 495 (2008).

[10] E. de Weerdt, E. van Kampen, D. van Gemert, Q. P. Chu, and J. A. Mulder, *Adaptive nonlinear dynamic inversion for spacecraft attitude control with fuel sloshing,* in *AIAA Guidance, Navigation and Control Conference and Exhibit* (2008).

[11] L. C. G. de Souza and A. G. de Souza, *Satellite attitude control system design considering the fuel slosh dynamics,* Shock and Vibration **2014** (2014).

[12] H. Zhang and Z. Wang, *Attitude control and sloshing suppression for liquid-filled spacecraft in the presence of sinusoidal disturbance,* Journal of Sound and Vibration **383**, 64 (2016).

[13] P. A. Mason and S. R. Starin, *Propellant slosh analysis for the solar dynamics observatory,* NASA technical report (2005).

[14] *International space station,* https://www.nasa.gov/mission_pages/station/main/index.html, photo taken in 2010.

[15] A. S. Saeed, A. B. Younes, S. Islam, J. Dias, L. Seneviratne, and G. Cai, *A review on the platform design, dynamic modeling and control of hybrid uavs,* in *2015 International Conference on Unmanned Aircraft Systems (ICUAS),* (IEEE, 2015) pp. 806–815.

[16] T. A. Weisshaar, *Morphing aircraft systems: historical perspectives and future challenges,* Journal of Aircraft (2013).

[17] R. M. Ajaj, C. S. Beaverstock, and M. I. Friswell, *Morphing aircraft: the need for a new design philosophy,* Aerospace Science and Technology **49**, 154 (2016).

[18] G. N. Saridis and H. E. Stephanou, *A hierarchical approach to the control of a prosthetic arm,* IEEE Transactions on Systems, Man, and Cybernetics **7**, 407 (1977).

[19] G. N. Saridis, *Analytic formulation of the principle of increasing precision with decreasing intelligence for intelligent machines,* Automatica **25**, 461 (1989).

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, *Reinforcement learning: A survey,* Journal of Artificial Intelligence Research **4**, 237 (1996).

[21] J. Kober, J. A. Bagnell, and J. Peters, *Reinforcement learning in robotics: A survey,* The International Journal of Robotics Research **32**, 1238 (2013).

[22] X. Xu, L. Zuo, and Z. Huang, *Reinforcement learning algorithms with function approximation: Recent advances and applications,* Information Sciences **261**, 1 (2014).

[23] L. Tang, M. Roemer, J. Ge, A. Crassidis, J. Prasad, and C. Belcastro, *Methodologies for adaptive flight envelope estimation and protection,* in *AIAA Guidance, Navigation, and Control Conference* (2009) p. 6260.

[24] E. R. Van Oort, L. Sonneveldt, Q. P. Chu, and J. A. Mulder, *Full-envelope modular adaptive control of a fighter aircraft using orthogonal least squares,* Journal of Guidance, Control, and Dynamics **33**, 1461 (2010).

[25] L. Sonneveldt, E. R. Van Oort, Q. P. Chu, and J. A. Mulder, *Nonlinear adaptive trajectory control applied to an F-16 model,* Journal of Guidance, Control, and Dynamics **32**, 25 (2009).

[26] J. Farrell, M. Sharma, and M. Polycarpou, *Backstepping-based flight control with adaptive function approximation,* Journal of Guidance, Control, and Dynamics **28**, 1089 (2005).

[27] L. Sonneveldt, E. R. Van Oort, Q. P. Chu, and J. A. Mulder, *Comparison of inverse optimal and tuning functions designs for adaptive missile control,* Journal of Guidance, Control, and Dynamics **31**, 1176 (2008).

[28] L. Sonneveldt, Q. P. Chu, and J. A. Mulder, *Nonlinear flight control design using constrained adaptive backstepping,* Journal of Guidance, Control, and Dynamics **30**, 322 (2007).

[29] R. Bellman, *Dynamic Programming* (Princeton University Press, 1957).

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, *Human-level control through deep reinforcement learning,* Nature **518**, 529 (2015).

[31] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning* (MIT Press, 1998).

[32] J. Si, *Handbook of learning and approximate dynamic programming,* Vol. 2 (John Wiley & Sons, 2004).

[33] F. L. Lewis and K. G. Vamvoudakis, *Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data,* IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, **41**, 14 (2011).

[34] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish, *Reinforcement learning and optimal adaptive control: An overview and implementation examples,* Annual Reviews in Control **36**, 42 (2012).

[35] S. Mahadevan and J. Connell, *Automatic programming of behavior-based robots using reinforcement learning,* Artificial Intelligencen **55**, 311 (1992).

[36] A. Brooks, A. Makarenko, S. Williams, and H. Durrant-Whyte, *Parametric POMDPs for planning in continuous state spaces,* Robotics and Autonomous Systems **54**, 887 (2006).

[37] A. Foka and P. Trahanias, *Real-time hierarchical pomdps for autonomous robot navigation,* Robotics and Autonomous Systems **55**, 561 (2007).

[38] A.-M. Zou and K. D. Kumar, *Quaternion-based distributed output feedback attitude coordination control for spacecraft formation flying,* Journal of Guidance, Control, and Dynamics **36**, 548 (2013).

[39] W. B. Powell, *Approximate dynamic programming: solving the curses of dimensionality* (John Wiley & Sons, 2007).

[40] V. R. Konda and J. N. Tsitsiklis, *Actor-critic algorithms,* in *Advances in Neural Information Processing Systems* (2000) pp. 1008–1014.

[41] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, *A survey of actor-critic reinforcement learning: Standard and natural policy gradients,* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**, 1291 (2012).

[42] D. P. Bertsekas, *Approximate policy iteration: A survey and some new methods,* Journal of Control Theory and Applications **9**, 310 (2011).

[43] F. L. Lewis and D. Vrabie, *Reinforcement learning and adaptive dynamic programming for feedback control,* IEEE Circuits and Systems Magazine **9** (2009).

[44] A. Keshavarz and S. Boyd, *Quadratic approximate dynamic programming for input-affine systems,* International Journal of Robust and Nonlinear Control **24**, 432 (2014).

[45] O. Sigaud and O. Buffet, *Markov decision processes in artificial intelligence* (John Wiley & Sons, 2013).

[46] Y. Zhou, E. van Kampen, and Q. P. Chu, *Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback,* Journal of Guidance, Control, and Dynamics **40**, 493 (2017), doi: http://arc.aiaa.org/doi/abs/10.2514/1.G001762.

[47] N. Szanto, V. Narayanan, and S. Jagannathan, *Event-sampled direct adaptive nn output-and state-feedback control of uncertain strict-feedback system,* IEEE Transactions on Neural Networks and Learning Systems (2017).

[48] Z. A. H. Scott A. Miller and E. K. P. Chong, *A POMDP framework for coordinated guidance of autonomous uavs for multitarget tracking,* EURASIP Journal on Advances in Signal Processing (2009).

[49] S. Ragi and E. K. P. Chong, *UAV path planning in a dynamic environment via partially observable markov decision process,* IEEE Transactions on Aerospace and Electronic Systems **49**, 2397 (2013).

[50] S. Ferrari and R. F. Stengel, *Online adaptive critic flight control,* Journal of Guidance, Control, and Dynamics **27**, 777 (2004).

[51] D. V. Prokhorov and D. C. Wunsch, *Adaptive critic designs,* IEEE Transactions on Neural Networks **8**, 997 (1997).

[52] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, *Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator,* IEEE Transactions on Neural Networks **13**, 764 (2002).

[53] R. Enns and J. Si, *Helicopter trimming and tracking control using direct neural dynamic programming,* IEEE Transactions on Neural Networks **14**, 929 (2003).

[54] E. van Kampen, Q. P. Chu, and J. A. Mulder, *Continuous adaptive critic flight control aided with approximated plant dynamics,* in *AIAA Guidance, Navigation, and Control Conference*, Vol. 5 (2006) pp. 2989–3016.

[55] J. Si and Y.-T. Wang, *Online learning control by association and reinforcement,* IEEE Transactions on Neural Networks **12**, 264 (2001).

[56] Z. Ni, H. He, X. Zhong, and D. V. Prokhorov, *Model-free dual heuristic dynamic programming,* IEEE Transactions on Neural Networks and Learning Systems **26**, 1834 (2015).

[57] Y. Wen, J. Si, X. Gao, S. Huang, and H. H. Huang, *A new powered lower limb prosthesis control framework based on adaptive dynamic programming,* IEEE transactions on neural networks and learning systems **28**, 2215 (2017).

[58] D. H. Wolpert and W. G. Macready, *No free lunch theorems for optimization,* IEEE Transactions on Evolutionary Computation **1**, 67 (1997).

[59] S. Takamuku and R. C. Arkin, *Multi-method learning and assimilation,* Robotics and Autonomous Systems **55**, 618 (2007).

[60] R. He, E. Brunskill, and N. Roy, *Efficient planning under uncertainty with macro-actions,* Journal of Artificial Intelligence Research **40**, 523 (2011).

[61] A. G. Barto and S. Mahadevan, *Recent advances in hierarchical reinforcement learning,* Discrete Event Dynamic Systems **13**, 41 (2003).

[62] M. Botvinick and A. Weinstein, *Model-based hierarchical reinforcement learning and human action control,* Philosophical Transactions of the Royal Society B **369**, 20130480 (2014).

[63] S. Sieberling, Q. P. Chu, and J. A. Mulder, *Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction,* Journal of Guidance, Control, and Dynamics **33**, 1732 (2010).

[64] P. Acquatella, W. Falkena, E. van Kampen, and Q. P. Chu, *Robust nonlinear spacecraft attitude control using incremental nonlinear dynamic inversion,* in *AIAA Guidance, Navigation, and Control Conference* (2012).

[65] P. Simplício, M. D. Pavel, E. van Kampen, and Q. P. Chu, *An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion,* Control Engineering Practice **21**, 1065 (2013).

[66] P. Lu, E. van Kampen, C. de Visser, and Q. P. Chu, *Aircraft fault-tolerant trajectory control using incremental nonlinear dynamic inversion,* Control Engineering Practice **57**, 126 (2016).

[67] P. Acquatella, E. van Kampen, and Q. P. Chu, *Incremental backstepping for robust nonlinear flight control,* in *Proceedings of the EuroGNC 2013* (2013).

[68] L. Sonneveldt, *Adaptive backstepping flight control for modern fighter aircraft* (Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 2010).

[69] S. H. Kim, Y. S. Kim, and C. Song, *A robust adaptive nonlinear control approach to missile autopilot design,* Control engineering practice **12**, 149 (2004).

[70] E. Todorov and W. Li, *A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems,* in *Proceedings of the 2005, American Control Conference, 2005.* (IEEE, 2005) pp. 300–306.

[71] D. Vrabie and F. Lewis, *Integral reinforcement learning for online computation of feedback nash strategies of nonzero-sum differential games,* in *49th IEEE Conference on Decision and Control (CDC)* (IEEE, 2010) pp. 3066–3071.

[72] J. Morimoto and C. G. Atkeson, *Minimax differential dynamic programming: An application to robust biped walking,* Advances in Neural Information Processing Systems 15 , 1539 (edited by Becker, S., Thrun, S., and Obermayer, K., MIT Press, Cambridge, MA, 2003).

[73] E. Bakolas and P. Tsiotras, *Feedback navigation in an uncertain flowfield and connections with pursuit strategies,* Journal of Guidance, Control, and Dynamics **35**, 1268 (2012).

[74] R. P. Anderson, E. Bakolas, D. Milutinović, and P. Tsiotras, *Optimal feedback guidance of a small aerial vehicle in a stochastic wind,* Journal of Guidance, Control, and Dynamics **36**, 975 (2013).

[75] Q. Hu, B. Jiang, and M. I. Friswell, *Robust saturated finite time output feedback attitude stabilization for rigid spacecraft,* Journal of Guidance, Control, and Dynamics **37**, 1914 (2014).

[76] S. Ulrich, J. Z. Sasiadek, and I. Barkana, *Nonlinear adaptive output feedback control of flexible-joint space manipulators with joint stiffness uncertainties,* Journal of Guidance, Control, and Dynamics **37**, 1961 (2014).

[77] F. Mazenc and O. Bernard, *Interval observers for linear time-invariant systems with disturbances,* Automatica **47**, 140 (2011).

[78] D. Efimov, T. Raïssi, S. Chebotarev, and A. Zolghadri, *Interval state observer for nonlinear time varying systems,* Automatica **49**, 200 (2013).

[79] M. R. Akella, D. Thakur, and F. Mazenc, *Partial lyapunov strictification: Smooth angular velocity observers for attitude tracking control,* Journal of Guidance, Control, and Dynamics **38**, 442 (2015).

[80] Y. Zhou, E. van Kampen, and Q. P. Chu, *Incremental approximate dynamic programming for nonlinear flight control design,* in *Proceedings of the 3rd CEAS EuroGNC: Specialist Conference on Guidance, Navigation and Control, Toulouse, France, 13-15 April 2015* (2015).

[81] Y. Zhou, E. van Kampen, and Q. P. Chu, *An incremental approximate dynamic programming flight controller based on output feedback,* in *AIAA Guidance, Navigation, and Control Conference* (2016) https://arc.aiaa.org/doi/abs/10.2514/6.2016-0360.

[82] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods* (Courier Corporation, 2007).

[83] M. Laban, *On-line aircraft aerodynamic model identification* (Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1994).

[84] Y. Zhou, E. van Kampen, and Q. P. Chu, *Adaptive spacecraft attitude control with incremental approximate dynamic programming,* in *Proceedings of the IAC 2017* (2017).

[85] D. Seo and M. R. Akella, *High-performance spacecraft adaptive attitude-tracking control through attracting-manifold design,* Journal of Guidance Control and Dynamics **31**, 884 (2008).

[86] A. Das, F. Lewis, and K. Subbarao, *Backstepping approach for controlling a quadrotor using lagrange form dynamics,* Journal of Intelligent and Robotic Systems **56**, 127 (2009).

[87] M. D. Tandale and J. Valasek, *Adaptive dynamic inversion control with actuator saturation constraints applied to tracking spacecraft maneuvers,* Journal of the Astronautical Sciences **52**, 517 (2004).

[88] K. A. Wise, E. Lavretsky, and N. Hovakimyan, *Adaptive control of flight: theory, applications, and open problems,* in *American Control Conference, 2006* (IEEE, 2006) pp. 6–pp.

[89] B. Kiumarsi, F. L. Lewis, M.-B. Naghibi-Sistani, and A. Karimpour, *Optimal tracking control of unknown discrete-time linear systems using input-output measured data,* IEEE transactions on cybernetics **45**, 2770 (2015).

[90] B. Açıkmeşe and S. R. Ploen, *Convex programming approach to powered descent guidance for mars landing,* Journal of Guidance, Control, and Dynamics **30**, 1353 (2007).

[91] Y. Zhou, E. van Kampen, and Q. P. Chu, *Incremental model based heuristic dynamic programming for nonlinear adaptive flight control,* in *Proceedings of the International Micro Air Vehicles Conference and Competition 2016, Beijing, China* (2016).

[92] R. Enns and J. Si, *Apache helicopter stabilization using neural dynamic programming,* Journal of Guidance Control and Dynamics **25**, 19 (2002).

[93] P. Singla, K. Subbarao, and J. L. Junkins, *Adaptive output feedback control for spacecraft rendezvous and docking under measurement uncertainty,* Journal of Guidance Control and Dynamics **29**, 892 (2006).

[94] S. D'Amico, J. S. Ardaens, G. Gaias, H. Benninghoff, B. Schlepp, and J. L. Jørgensen, *Noncooperative rendezvous using angles-only optical navigation: system design and flight results,* Journal of Guidance, Control, and Dynamics (2013).

[95] B. Dachwald, H. Boehnhardt, U. Broj, U. R. Geppert, J.-T. Grundmann, W. Seboldt, P. Seefeldt, P. Spietz, L. Johnson, E. Kührt, *et al.*, *Gossamer roadmap technology reference study for a multiple neo rendezvous mission,* in *Advances in Solar Sailing* (Springer, 2014) pp. 211–226.

[96] *Automated transfer vehicle,* https://en.wikipedia.org/wiki/Automated_Transfer_Vehicle (Acessed: 2017-11-19).

[97] *Automated transfer vehicle,* https://www.nasa.gov/mission_pages/station/structure/atv.html (Acessed: 2017-11-10).

[98] B. Luo, D. Liu, T. Huang, and D. Wang, *Model-free optimal tracking control via critic-only q-learning,* IEEE transactions on neural networks and learning systems **27**, 2134 (2016).

[99] S. Haykin, *Adaptive filter theory* (Prentice Hall, 2002).

[100] Y. Zhou, E. van Kampen, and Q. P. Chu, *Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming,* in *Proceedings of the IAC 2017* (2017).

[101] T. Hanselmann, L. Noakes, and A. Zaknich, *Continuous-time adaptive critics,* IEEE Transactions on Neural Networks **18**, 631 (2007).

[102] V. Yadav, R. Padhi, and S. Balakrishnan, *Robust/optimal temperature profile control of a high-speed aerospace vehicle using neural networks,* IEEE Transactions on Neural Networks **18**, 1115 (2007).

[103] F. Y. Wang, H. Zhang, and D. Liu, *Adaptive dynamic programming: an introduction,* Computational Intelligence Magazine, IEEE **4**, 39 (2009).

[104] L. Dong, X. Zhong, C. Sun, and H. He, *Adaptive event-triggered control based on heuristic dynamic programming for nonlinear discrete-time systems,* IEEE transactions on neural networks and learning systems (2017).

[105] S. Haykin, *Neural networks: a comprehensive foundation* (Prentice Hall International, Inc., 1999).

[106] Y. Zhou, E. van Kampen, and Q. P. Chu, *Incremental model based online dual heuristic programming for nonlinear adaptive control,* Control Engineering Practice **73**, 13 (2018), https://doi.org/10.1016/j.conengprac.2017.12.011.

[107] D. Wang, H. He, and D. Liu, *Adaptive critic nonlinear robust control: A survey,* IEEE Transactions on Cybernetics **47**, 3429 (2017).

[108] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, *Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming,* Automatica **48**, 1825 (2012).

[109] I. E. Putro and F. Holzapfel, *Robust flight control design using incremental adaptive sliding mode control,* in *2016 International Conference on Instrumentation, Control and Automation (ICA)* (IEEE, 2016) pp. 114–119.

[110] G. Looye and H.-D. Joos, *Design of robust dynamic inversion control laws using multi-objective optimization,* in *Proceedings of the AIAA Guidance, Navigation and Control Conference* (2001).

[111] P. Lu, L. Van Eykeren, E. van Kampen, C. de Visser, and Q. Chu, *Double-model adaptive fault detection and diagnosis applied to real flight data,* Control Engineering Practice **36**, 39 (2015).

[112] T. Lombaerts, G. Looye, Q. P. Chu, and J. A. Mulder, *Design and simulation of fault tolerant flight control based on a physical approach,* Aerospace Science and Technology **23**, 151 (2012).

[113] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, *A survey of multi-objective sequential decision-making,* Journal of Artificial Intelligence Research **48**, 67 (2013).

[114] C. Liu, X. Xu, and D. Hu, *Multiobjective reinforcement learning: A comprehensive overview,* IEEE Transactions on Systems, Man, and Cybernetics: Systems **45**, 385 (2015).

[115] K. Van Moffaert and A. Nowé, *Multi-objective reinforcement learning using sets of pareto dominating policies,* The Journal of Machine Learning Research **15**, 3483 (2014).

[116] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, *Empirical evaluation methods for multiobjective reinforcement learning algorithms,* Machine learning **84**, 51 (2011).

[117] I. Y. Kim and O. De Weck, *Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation,* Structural and multidisciplinary optimization **31**, 105 (2006).

[118] J. G. Lin, *On min-norm and min-max methods of multi-objective optimization,* Mathematical programming **103**, 1 (2005).

[119] W. S. Lovejoy, *A survey of algorithmic methods for partially observed markov decision processes,* Annals of Operations Research **28**, 47 (1991).

[120] J. Hoey, T. Schröder, and A. Alhothali, *Affect control processes: Intelligent affective interaction using a partially observable markov decision process,* Artificial Intelligence **230**, 134 (2016).

[121] G. Baldassarre and M. Mirolli, eds., *Computational and robotic models of the hierarchical organization of behavior* (Springer, 2013).

[122] R. E. Parr and S. Russell, *Hierarchical control and learning for markov decision processes* (University of California, Berkeley Berkeley, CA, 1998).

[123] R. Parr and S. Russell, *Reinforcement learning with hierarchies of machines,* Advances in neural information processing systems , 1043 (1998).

[124] R. S. Sutton, D. Precup, and S. Singh, *Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,* Artificial intelligence **112**, 181 (1999).

[125] T. G. Dietterich, *Hierarchical reinforcement learning with the maxq value function decomposition,* Journal of Artificial Intelligence Research (JAIR) **13**, 227 (2000).

[126] M. Ghavamzadeh, S. Mahadevan, and R. Makar, *Hierarchical multi-agent reinforcement learning,* Autonomous Agents and Multi-Agent Systems **13**, 197 (2006).

[127] B. Hengst, *Discovering hierarchy in reinforcement learning with hexq,* in *International Conference on Machine Learning (ICML)*, Vol. 2 (2002) pp. 243–250.

[128] G. Theocharous, K. Rohanimanesh, and S. Maharlevan, *Learning hierarchical observable markov decision process models for robot navigation,* in *IEEE International Conference on Robotics and Automation, ICRA'01*, Vol. 1 (IEEE, 2001) pp. 511–516.

[129] G. Theocharous and S. Mahadevan, *Approximate planning with hierarchical partially observable markov decision process models for robot navigation,* in *IEEE International Conference on Robotics and Automation, ICRA'02.*, Vol. 2 (IEEE, 2002) pp. 1347–1352.

[130] M. Sridharan, J. Wyatt, and R. Dearden, *Planning to see: A hierarchical approach to planning visual actions on a robot using pomdps,* Artificial Intelligence **174**, 704 (2010).

[131] Y. Zhou, E. van Kampen, and Q. P. Chu, *Autonomous navigation in partially observable environments using hierarchical q-learning,* in *Proceedings of the International Micro Air Vehicles Conference and Competition 2016, Beijing, China* (2016).

[132] S. J. Pan and Q. Yang, *A survey on transfer learning,* IEEE Transactions on knowledge and data engineering **22**, 1345 (2010).

[133] R. Caruana, *Multitask learning,* in *Learning to learn* (Springer, 1998) pp. 95–133.

[134] M. I. Jordan and T. M. Mitchell, *Machine learning: Trends, perspectives, and prospects,* Science **349**, 255 (2015).

[135] L. Shao, F. Zhu, and X. Li, *Transfer learning for visual categorization: A survey,* IEEE transactions on neural networks and learning systems **26**, 1019 (2015).

[136] H. H. Zhuo and Q. Yang, *Action-model acquisition for planning via transfer learning,* Artificial intelligence **212**, 80 (2014).

[137] M. E. Taylor, P. Stone, and Y. Liu, *Transfer learning via inter-task mappings for temporal difference learning,* Journal of Machine Learning Research **8**, 2125 (2007).

[138] M. E. Taylor and P. Stone, *Transfer learning for reinforcement learning domains: A survey,* Journal of Machine Learning Research **10**, 1633 (2009).

[139] A. Lazaric, *Transfer in reinforcement learning: a framework and a survey,* Reinforcement Learning - State of the art **12**, 143 (2012), springer.

[140] B. Fernandez-Gauna, J. M. Lopez-Guede, and M. Graña, *Transfer learning with partially constrained models: application to reinforcement learning of linked multicomponent robot system control,* Robotics and Autonomous Systems **61**, 694 (2013).

[141] D. Rasmussen, A. Voelker, and C. Eliasmith, *A neural model of hierarchical reinforcement learning,* PLOS one **12(7)**, e0180234 (2017).

[142] M. Abdoos, N. Mozayani, and A. L. Bazzan, *Traffic light control in non-stationary environments based on multi agent q-learning,* in *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (IEEE, 2011) pp. 1580–1585.

[143] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, *Learning in nonstationary environments: A survey,* IEEE Computational Intelligence Magazine **10**, 12 (2015).

[144] M. Leonetti, L. Iocchi, and P. Stone, *A synthesis of automated planning and reinforcement learning for efficient, robust decision-making,* Artificial Intelligence **241**, 103 (2016).

[145] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, *Simultaneous localization and mapping: A survey of current trends in autonomous driving,* IEEE Transactions on Intelligent Vehicles **2**, 194 (2017).

[146] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, *Simultaneous localization, mapping and moving object tracking,* The International Journal of Robotics Research **26**, 889 (2007).

[147] A. Kawewong, N. Tongprasit, S. Tangruamsub, and O. Hasegawa, *Online and incremental appearance-based slam in highly dynamic environments,* The International Journal of Robotics Research **30**, 33 (2011).

[148] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, *Dynamic pose graph slam: Long-term mapping in low dynamic environments,* in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2012) pp. 1871–1878.

[149] M. Rida, H. Mouncif, and A. Boulmakoul, *Application of markov decision processes for modeling and optimization of decision-making within a container port,* in *Soft Computing in Industrial Applications* (Springer, 2011) pp. 349–358.

[150] C. J. Watkins and P. Dayan, *Q-learning,* Machine learning **8**, 279 (1992).

[151] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, 1st ed. (Athena Scientific, 1996).

[152] C. C. De Visser, Q. P. Chu, and J. A. Mulder, *A new approach to linear regression with multivariate splines,* Automatica **45**, 2903 (2009).

[153] C. Lesiak and A. Krener, *The existence and uniqueness of volterra series for nonlinear systems,* IEEE transactions on automatic control **23**, 1090 (1978).

[154] W. Guo, J. Si, F. Liu, and S. Mei, *Policy approximation in policy iteration approximate dynamic programming for discrete-time nonlinear systems,* IEEE Transactions on Neural Networks and Learning Systems (2017).

Samenvatting

Online Reinforcement Learning voor Lucht- en Ruimtevaartsystemen

Ye ZHOU

Recente technologische ontwikkelingen hebben geleid tot ontwikkeling van innovatieve en geavanceerde lucht- en ruimtevaartsystemen. De toename van complexiteit van deze systemen is de grootste uitdaging geworden voor het ontwerp van regelsystemen in de lucht- en ruimtevaart. Multidisciplinaire taken in toepassingen variërend van het luchtvaartdomein tot het ruimtevaartdomein, van commerciëel tot militair, vergroten de complexiteit en de eisen die gesteld worden aan automatische besturingssystemen. Bovendien hebben onzekerheden in lucht- en ruimtevaartsystemen, zoals in vliegtuigen met aanpasbare vleugelvorm, alsook onzekerheden in de omgeving, zoals plotselinge windstoten, complex luchtverkeer en de impact van ruimtepuin, de noodzaak tot het zichzelf online aanpassen van regelsystemen vergroot. Om overweg te kunnen met deze groeiende complexiteit van systeemdynamica, met de toename van moeilijkheidsgraad van besturingstaken en met de eis tot aanpasbaarheid, is er voor lucht- en ruimtevaartsystemen een dringende noodzaak voor hogere niveaus van autonomie.

De complexiteit en diversiteit van lucht- en ruimtevaartsystemen en autonome besturingstaken hebben onderzoekers gemotiveerd tot het onderzoeken van intelligente methodes. Intelligente autonome lucht- en ruimtevaartsystemen dienen aan de ene kant de huidige systeemdynamica en omgeving online te leren om zichzelf daarop aan te passen en om dit nauwkeurig te besturen. Aan de ander kant moeten deze systemen ook een afweging kunnen maken tussen meerdere doelen en de veiligheid waarborgen. Om deze reden hebben intelligente systemen vaak een hiërarchische besturingsstructuur.

Low-level besturing is de basis voor de hogere besturingsniveaus en limiteert de prestaties van het gehele autonome besturingssysteem. Deze beperking is de belangrijkste reden dat veel van de bestaande *high-level* algoritmes nog niet succesvol geïmplementeerd kunnen worden op echte lucht- en ruimtevaartsystemen. Bovendien moet de intelligentie en autonomie van *high-level* besturingssystemen verbeterd worden om nieuwe uitdagingen aan te gaan in lucht- en ruimtevaartsystemen, zoals deep-space onderzoek, navigatie van drones binnen gebouwen of zichzelf organiserende zwermen van onbemande voertuigen.

Reinforcement Learning (RL) is een intelligente, zelf-lerende methode die toegepast kan worden op verschillende niveaus van autonome operaties. Het linkt op de natuur geïnspireerde kunstmatige intelligentie met het vakgebied van regeltheorie. RL technieken kunnen voor low-level besturing worden gebruikt om de efficiëntie en zelf-aanpasbaarheid van het regelsysteem te vergroten wanneer het dynamische model onzeker of onbekend is. Low-level besturingstaken, zoals stabilisatie en volgtaken, worden vaak gemodelleerd in het domein van continue acties en toestandsvariabelen. Op een hoger niveau kan RL gebruikt worden om de intelligentie van beslissingen en planning te vergroten en om coördinatie met de lagere besturingsniveaus te waarborgen. Op dit hogere niveau kunnen de acties en toestandsvariabelen discreet of continue zijn, of zelfs een hybride mix van deze twee.

RL methodes zijn relatief nieuw in het gebied van besturing en navigatie van lucht- en ruimtevaartsystemen. Ze hebben bij toepassing in dit gebied veel voordelen, maar ook enkele beperkingen. Dit proefschrift heeft als doel om de volgende onderzoeksvraag te beantwoorden:

> **Hoe kan een lucht- en ruimtevaartsysteem RL methodes toepassen om de autonomie van het online leren te verbeteren wanneer de omgeving initiëel onbekend is, er onzekerheden zijn in het dynamische model en wanneer het systeem niet volledig observeerbaar is?**

Deze hoofdvraag wordt behandeld in drie delen, voor drie specifieke RL methodes en toepassingen: (i) Approximate Dynamic Programming (ADP) voor besturingstaken met een bij benadering convexe cost-to-go functie, (ii) Adaptive Critic Designs (ACDs) voor algemene niet-lineaire regelproblemen, en (iii) Hierarchical Reinforcement Learning (HRL) voor high-level navigatie. Dit leidt tot de volgende drie onderzoeksvragen:

1. Hoe kan Linear Approximate Dynamic Programming (LADP) gegeneraliseerd worden om overweg te kunnen met niet-lineaire en/of tijd-variërende systemen, modelfouten en onvolledige observeerbaarheid, terwijl de efficiëntie en de expliciete mathematische uitdrukking behouden blijft?

2. Hoe moeten Adaptive Critic Designs ontworpen worden om de online zelf-leerbaarheid te verbeteren onder invloed van interne onzekerheden, externe verstoringen en zelfs plotselinge fouten in het systeem?

3. Hoe kan op een systematische wijze een Hierarchical Reinforcement Learning regelaar ontwikkeld worden die werkt met meerdere doelen, onvolledige observeerbaarheid van het systeem, die de mogelijkheid heeft om geleerd gedrag over te dragen en die meerdere RL technieken toepast?

Om de eerste vraag te beantwoorden worden in dit proefschrift *incremental* Approximate Dynamic Programming (iADP) methodes voorgesteld. In plaats van niet-lineaire functie-benaderingen voor de cost-to-go gebruiken iADP methodes een incrementeel model om om te gaan met onbekende niet-lineaire systemen en onzekerheden in de omgeving. Deze methodes kunnen nog steeds een kwadratische kostfunctie toepassen om een efficiënt en mathematisch expliciet *optimal control* algoritme te genereren. Ze hebben geen a-priori kennis van de systeemdynamica nodig, geen online identificatie van het globale model, noch een aanname over tijdschaal separatie, maar slechts een online geïdentificeerd incrementeel model.

Als eerste wordt voorgesteld om een regulerings-probleem voor niet-lineaire systemen op te lossen met iADP's. Als de meting van de volledige toestand van het systeem bekend is, dan kan een incrementeel model geïdentificeerd worden om de volgende toestand van het systeem te voorspellen. Met deze voorspelling en een kwadratische kostfunctie kan een incrementele toename van de uitvoer van de regelaar berekend worden die voldoet aan de principes van optimal control. Als alleen in invoer en uitvoer van het dynamische systeem bekend zijn, dan wordt de optimale incrementele toename van de regelaar berekend met een output-feedback algoritme en een incrementeel model. Deze methode is toegepast om iteratief de vliegbesturing van een niet-lineair raketmodel te optimaliseren, zowel met volledige toestandskennis als met metingen van de systeemuitvoer. Simulatieresultaten tonen aan dat de iADP methode de prestaties van de regelaar voor het niet-lineaire systeem verbeteren, terwijl het ontwerpproces simpel en systematisch is.

Het concept van iADP is verder uitgebreid naar volgtaken voor niet-lineaire Multiple-Input Multiple-Output (MIMO) systemen en voor systemen met onvolledige observeerbaarheid. Omdat iADP methodes een aparte structuur hebben om de lokale systeemdynamica te representeren, kan de kostfunctie minder afhankelijk zijn van het systeem of de referentie-invoer en hoeft deze slechts een ruwe benadering van de cost-to-go te zijn. Deze benadering is een kwadratische functie van alleen de huidige volgfout, zonder de dimensie van de systeemtoestand voor de kostfunctie aan te passen naar een aangevulde versie.

Voor de volgtaak worden twee condities in acht genomen voor de observeerbaarheid van het systeem. Wanneer een directe meting van de gehele toestand van het systeem beschikbaar is, dan wordt het incrementele model online geïdentificeerd om zo de optimale incrementele toename van de uitvoer van de regelaar te ontwerpen. Als de volgfout voor het volgen van een stochastische dynamische referentie de enige meting is, dan wordt het systeem voor een deel onobserveerbaar. De observaties worden gebruikt om het incrementele model te identificeren en om de opvolgende volgfout van de systeemuitvoer te schatten voor de optimal-control volgtaak.

Voor beide condities van observeerbaarheid is een offline algoritme toegepast om de regelaar iteratief te verbeteren totdat de nauwkeurigheid groot genoeg is. Vervolgens wordt een online algoritme gebruikt om de regelaar recursief verder te trainen. Deze recursieve algoritmes kunnen ook online gebruikt worden in echte systemen die mogelijk verschillen van het systeemmodel dat gebruikt was in de iteratieve offline leerfase. Deze algoritmes zijn toegepast op een standsbesturingstaak van een gesimuleerde satelliet die verstoord wordt door het klotsen van interne vloeistof. De resultaten laten zien dat het voorgestelde algoritme nauwkeurig en adaptief omgaat met tijdsvariërende interne dynamica, terwijl de besturing efficiënt blijft, in het bijzonder voor onbekende niet-lineaire systemen met onvolledige observeerbaarheid.

Om de tweede onderzoeksvraag te beantwoorden ontwikkelt dit proefschrift online ACD's gebaseerd op het *incrementele model.* ACD's kunnen in het algemeen gecategoriseerd worden in drie groepen: 1) Heuristic Dynamic Programming (HDP), 2) Dual Heuristic Programming (DHP), en 3) Globalized Dual Heuristic Programming (GDHP). Van deze drie versies zijn ook actie-afhankelijke versies ontwikkeld door middel van een directe connectie tussen de uitvoer van de Actor met de invoer van de Critic. Dit proefschrift richt zich op actie-*on*afhankelijke ACD's, specifiek HDP en DHP.

Een HDP methode gebaseerd op een incrementeel model wordt geïntroduceerd, IHDP, om online en adaptief onbekende lucht- en ruimtevaartsystemen te besturen. Hierbij wordt de benadering van het globale model vervangen door een icrementeel model. Deze aanpak heeft daardoor geen offline trainingsfase nodig en kan het online leren versnellen. De IHDP methode wordt vergeleken met de conventionele HDP methode voor een online volgtaak van een onbekend niet-lineair raketmodel. De resultaten laten zien dat de voorgestelde IHDP methode het online leren versneld, dat er een hogere nauwkeurigheid is in de volgtaak en dat er omgegaan kan worden met een grotere variatie in initiële condities dan de conventionele HDP methode. Daarnaast is de IHDP methode toegepast op een MIMO standsregelaar voor een satelliet die verstoord wordt door klotsen van interne vloeistof en die externe verstoringen heeft. De simulatieresultaten demonstreren dat de IHDP methode adaptief en robuust is voor interne onzekerheden en externe verstoringen.

Om de prestaties van de regelaar verder te verbeteren en om het online leren te versnellen is een Dual Heuristic Programming methode ontwikkeld gebaseerd op een incrementeel model: IDHP. IDHP gebruikt een recursieve kleinste-kwadraten methode (RLS) om realtime het incrementele model te identificeren in plaats van het globale model. Naast een online volgtaak wordt ook een Fault-Tolerant Control (FTC) taak uitgevoerd, zowel met IDHP als met de conventionele DHP. De resultaten demonstreren dat de IDHP methode met succes een instabiel systeem met fouten kan besturen voordat de toestand van het systeem divergeert, terwijl DHP hierin faalt. Om de robuustheid van de voorgestelde IDHP methode verder te valideren is hoogfrequente meetruis toegevoegd aan de metingen van de toestandsvariabelen. De simulatieresultaten tonen dat de IDHP methode niet gevoelig is voor deze meetruis.

De derde onderzoeksvraag is beantwoord door de ontwikkeling van een *hybride* Hierarchical Reinforcement Learning (hHRL) methode voor navigatieproblemen. Deze methode bestaat uit een aantal hiërarchische niveaus, waarbij elk niveau een andere methode kan gebruiken om te leren, met verschillende soorten van informatie en verschillende doelen. Er worden expliciete regels geformuleerd voor het opzetten van de hiërarchie, voor het opsplitsen van de taken en voor het toekennen van de beloningen. Gedetailleerde implementaties van de voorgestelde hHRL methode zijn gepresenteerd voor een online navigatie taak met onvolledige observeerbaarheid en meerdere doelen (bijv. het naderen van een doel terwijl obstakels ontweken moeten worden).

De voorgestelde methode is ten eerste toegepast op een doolhof-probleem uit de literatuur, waarbij botsingen voorkomen moeten worden en de prestatie van het bereiken van de doelstaat online en per episode verbeterd worden. Het resultaat wordt vergeleken met een 'platte' RL implementatie en met een HRL implementatie die slechts een enkele leervorm gebruikt. De resultaten tonen dat hHRL efficiënter omgaat met de 'curse of dimensionality' en dat onzekerheid en ambiguïteit verkleind worden.

De geleerde resultaten worden daarna toegepast op een groter doolhof, wat valideert dat de geleerde resultaten overgebracht kunnen worden naar nieuwe taken waardoor het leren van nieuwe taken of in nieuwe omgevingen sneller wordt. Tot slot wordt dezelfde methode toegepast op een niet-stationaire omgeving met aangepaste sensoren en een gedeeltelijke plattegrond. De hHRL methode kan, zonder verlies van efficiëntie, leren in niet-stationaire omgevingen door relatieve micro-toestandsvariabelen te gebruiken en absolute macro-toestandsvariabelen. Deze resultaten laten zien dat de voorgestelde hHRL methode kan helpen bij het versnellen van leren, bij het verminderen van het effect van de 'curse of dimensionality' in complexe taken, bij het verminderen van de onzekerheid en ambiguïteit, bij het efficiënt overzetten van geleerd gedrag naar nieuwe taken en bij implementatie in niet-stationaire omgevingen. De voorgestelde methode heeft de potentie om een bijna optimaal hiërarchisch gedrag te leren voor een navigatietaak met een onbekend systeem in een onbekende omgeving.

Concluderend kan gesteld worden dat dit proefschrift een bijdrage levert met verschillende methodes die de intelligentie en autonomie van lucht- en ruimtevaartsystemen verbeteren. Deze verbeteringen komen voort uit drie perspectieven: 1) het verbeteren van de aanpasbaarheid en efficiëntie van low-level besturing, 2) het verbeteren van intelligentie en het online leren van besturing en navigatie methodes, 3) het creëren van een goed georganiseerde hiërarchie om coördinatie tussen de verschillende niveaus te waarborgen. De voorgestelde methodes bieden nieuwe inzichten voor zowel de Reinforcement Learning gemeenschap als voor ontwikkelaars van automatische besturingssystemen in de lucht- en ruimtevaart.

Acknowledgements

This dissertation is the result of my four and a half years of research in Control and Simulation (C&S) division at the Faculty of Aerospace Engineering, Delft University of Technology. It would not be possible to finish this work without the support of many people that I met here in Delft. At the end of this dissertation and also of my PhD journey, I would like to thank all of these people and to mention some of them in more detail.

First of all, I would like to thank my promoter, Prof. Max Mulder, for your guidance and also support from other aspects. I am impressed by your kindness to people, conscientiousness to work, and efficiency in giving me feedback. Without any of these, my dissertation would not be here as what it is.

To Dr. Qi Ping Chu, my second promoter, I feel deep gratitude and respect, not only for your academic insights, valuable suggestions, and enthusiastic encouragement during my research, but also for your kind help and concern for me and my family during our most difficult period. You encouraged me to work on my own ideas and guided me with valuable advice to make them feasible.

To Dr. Erik-Jan van Kampen, my co-promoter and daily supervisor, I would like to express my very sincere gratitude and admiration for your patient guidance and effective supervision. You are always willing to offer me valuable and constructive suggestions and to share knowledge with me so generously during my research work. Every discussion with you is beneficial for me. On the other hand, you give me enough freedom to pursue my research and provide me a relaxed research experience, which is very much appreciated.

Many thanks to Dr. Coen de Visser, who was the first person I met in C&S and treated me an unforgettable Dutch-style lunch. My special thanks to Dr. Daan Pool, who offered me and my family generous help during our stay in the Netherlands. Bertine Markus offered her valuable support in all the administrative works and also in other aspects, which I really appreciate.

My thanks also goes out to all the colleagues in C&S. Tommaso Mannucci, Jaime Junell, Kirk Scheper, and Erik-Jan van Kampen, thank you for making our IFC group meetings very great and pleasant for exchanging our knowledge and ideas. I learned a lot from you. My former and current 0.04 roommates, Dyah Jatiningrum, João Caetano, Tommaso Mannucci, Emmanuel Sunil, Peng Lu, Jan Smisek, Annemarie Landman, Sihao Sun, Shuo Li, Shushuai Li, Ying Yu, and Tom van Dijk, thank you for making this room so nice and special to me. I would like to express my very warm thanks to Dyah Jatiningrum, Marilena Pavel, Junzi Sun&Marie Kummerlowe, Ye Zhang&Yingzhi Huang, Sherry Wang, Sihao Sun for your support for me and my family. My gratitude are extended to Bob Mulder, Jacco Hoekstra, René van Paassen, Guido de Croon, Olaf Stroosma, Clark Borst, Alexander in 't Veld, Hans Mulder, Joost Ellerbroek, Bart Remes, Christophe de Wagter, Erik van der Horst, Ferdinand Postema, Andries Muis, Harold Thung, Alwin Damman, Menno Klaassen, Matěj Karásek, Maarten Tielrooij, Rolf Klomp, Gustavo Mercado Velasco, Jan Comans, Deniz Yilmaz, Jia Wan, Liguo Sun, Laurens van Eykeren, Yazdi Jenie, Sjoerd Tijmons, Sophie Armanini, Wei Fu, Lei Yang, Tao Lu, Henry Tol, Ewoud Smeur, Tim Visser, Ivan Miletovic, Kasper van der El, Jerom Maas, Isabel Metz, Julia Rudnyk, Dirk van Baelen, Sarah Barendswaard, Kimberly Mcguire, Kevin van Hecke, Elisabeth van der Sman, Roland Meertens, Mario Coppola, Daniel Friesen, Ezgi Akel, and all other colleagues I met in C&S. Thank you all for making C&S such a nice place.

Many people outside C&S have been also very important to me during the last few years. I would like to thank my former supervisor, Prof. Wei Zhang, who brought me to the world of Aerospace Engineering. My special thanks to Mrs. Jianying Zhang for all your support and kind help to me and to my family. My gratitude are extended to Xiaoxia Yang, Jie Zhou, Changjie Zhan, Zhipei Chen, Yannian Yang, Peijian Lv, Zaoxu Zhu, Fengnian Tian, Shuanghou Deng, Wei Yu, Jing Dong, Duo Zou, Xu Ma, and all other friends I met in the Netherlands. I am so glad to have met you.

I gratefully acknowledge the China Scholarship Council for their four-year funding and all the other support.

My profound gratitude goes out to my parents, Xu He and Kai Zhou, for your unconditional love and support for me and for my present family. You raised me to be happy, healthy, and loved. No matter what path I choose, you always give me liberty, believe in me, and support me throughout my life. I would also like to thank my great-grandmother, my grandparents, my uncle and aunt, and all my families for your support and love. My thanks also extended to my family-in-law, which gave me a chance to enjoy a warm atmosphere in a very big family.

Finally, my deepest gratitude goes to my beloved husband, Hann Woei Ho, for your continued support, understanding, encouragement, and unfailing love. You were always there whenever I need you. I greatly appreciate your contribution and your belief in me. I also appreciate my little girl, Zi Xuan Ho, for giving me so much happiness and abiding my ignorance when I am busy. I hope I have been a good mother and will be better. I feel so grateful to have both of you.

Curriculum Vitæ

Ye ZHOU

13-10-1988	Born in Hefei, Anhui, China.

Education

2006–2009	Honors College Northwestern Polytechnical University (NPU), China
2009–2010	BSc, School of Mechanical & Electrical Engineering Northwestern Polytechnical University (NPU), China
2010–2013	MSc, School of Mechanical & Electrical Engineering Northwestern Polytechnical University (NPU), China
2012–2013	School of Aeronautics Northwestern Polytechnical University (NPU), China
2013–2018	Ph.D., Faculty of Aerospace Engineering Delft University of Technology (TU Delft), the Netherlands *Thesis:* Online Reinforcement Learning Control for Aerospace Systems *Promotor:* Prof. dr. ir. M. Mulder and dr. Q. P. Chu

Awards

2007	Distinguished Bachelor student (in NPU)
2010	Distinguished Bachelor graduation project in industrial design (national)
2010	Distinguished Bachelor degree dissertation (in NPU)
2011	Distinguished Master student (in NPU)
2013	Distinguished Master degree dissertation (in NPU)

PUBLICATIONS DURING MASTER STAGE

1. B. Gou, **Y. Zhou**, S. Yu, et al., *Research on Semantic-driven Intelligent Color Design based on DNN*, International Proceedings of Computer Science and Information Technology (IPCSIT),2011.

2. Z. Ma, W. Zhang, **Y. Zhou**, et al., *The Glare Evaluation Method Using Digital Camera for Civil Airplane Flight Deck*, Engineering Psychology and Cognitive Ergonomics. Applications and Services, Springer Berlin Heidelberg, 2013, 184-192.

3. **Y. Zhou**, W. Zhang, B. Li, et al., *A coherent assessment of visual ergonomics in flight deck impacted by color and luminance*, Engineering Psychology and Cognitive Ergonomics. Applications and Services, Springer Berlin Heidelberg, 2013, 222-230.

4. **Y. Zhou**, S. Yu, J. Chu, *Research on Semantic-driven Intelligent Color Design based on Dynamic Fuzzy Neural Network*, Computer Engineering and Application, 50 (3), 2014, in Chinese

SOFTWARE COPYRIGHT

1. **Y. Zhou**, S. Yu, *Semantic-driven Intelligent Color Design System*, P. R. China Software Copyright 2011SR043559, filed June 2010 and issued July 2011.

2. W. Wang, J. Chu, **Y. Zhou**, *Design and Assessment Software of Human-machine System*, P. R. China Software Copyright 2011SR043478, filed June 2010 and issued July 2011.

List of Publications

Journals

5. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Hybrid hierarchical reinforcement learning with partial observability*, Artificial Intelligence (submitted).

4. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Incremental approximate dynamic programming for nonlinear adaptive tracking control with partial observability*, Journal of Guidance, Control, and Dynamics, (under review).

3. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming*, Acta Astronautica, (under review).

2. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Incremental model based online dual heuristic programming for nonlinear adaptive control*, Control Engineering Practice, Vol. 73, p. 13-25, 2018. https://doi.org/10.1016/j.conengprac.2017.12.011

1. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback*, Journal of Guidance, Control, and Dynamics, Vol. 40, No. 2, p. 493-500, 2017. https://doi.org/10.2514/1.G001762

Conference proceedings

7. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Adaptive spacecraft attitude control with incremental approximate dynamic programming*, in 68th International Astronautical Congress (IAC) (Adelaide, Australia, 2017).

6. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming*, in 68th International Astronautical Congress (IAC) (Adelaide, Australia, 2017).

5. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Autonomous navigation in partially observable environments using hierarchical Q-Learning*, in International Micro Air Vehicle Conference and Competition 2016 (IMAV 2016) (Beijing, PR of China, 2016) p. 70-76.

4. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Incremental model based heuristic dynamic programming for nonlinear adaptive flight control*, in International Micro Air Vehicle Conference and Competition 2016 (IMAV 2016) (Beijing, PR of China, 2016) p. 173-180.

3. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *An incremental approximate dynamic programming flight controller based on output feedback*, AIAA Guidance, Navigation, and Control Conference (San Diego, California, USA, 2016) (AIAA 2016-0360). https://doi.org/10.2514/6.2016-0360

2. J. Junell, T. Mannucci, **Y. Zhou**, and E. van Kampen, *Self-tuning Gains of a Quadrotor using a Simple Model for Policy Gradient Reinforcement Learning*, AIAA Guidance, Navigation, and Control Conference (San Diego, California, USA, 2016) (AIAA 2016-1387). https://doi.org/10.2514/6.2016-1387

1. **Y. Zhou**, E. van Kampen, and Q. P. Chu, *Incremental approximate dynamic programming for nonlinear flight control design*, in 3rd CEAS EuroGNC conference (Toulouse, France, 2015)