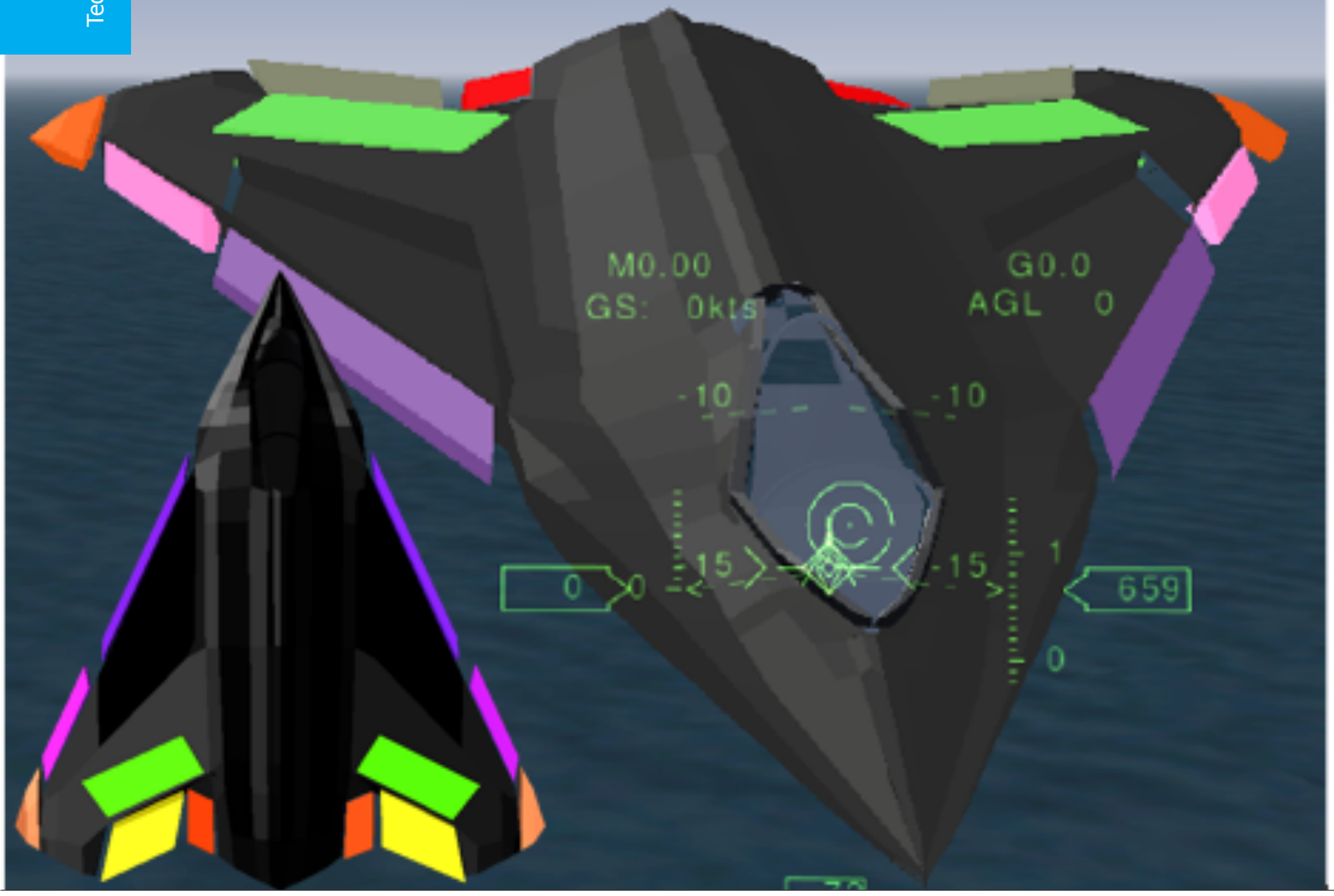
MSc Thesis Report

Application of Continuous Reinforcement Learning on Innovative Control Effector Aircraft

K. Shayan

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

MSc Thesis Report

Application of Continuous Reinforcement Learning on Innovative Control Effector Aircraft

by

K. Shayan

In fulfillment of the requirements for the MSc Thesis Project as an essential part of master graduation.

Master of Science
Aerospace Engineering

at the Delft University of Technology,
to be presented on October, 2019.

Supervisor: Dr. Ir. E. van Kampen

TU Delft Delft University of Technology

Summary

Objective: This study aims to design a continuous reinforcement learning-based controller for the tailless Innovative Control Effector (ICE) aircraft considering a predefined objective (e.g., attitude control). The over-actuated configuration and control constraints for ICE aircraft innovation design are introduced to reduce detection signatures and increase survivability while maintaining the high maneuverability. Besides, the next generation of fighters, including ICE, is aimed to be completely autonomous. The challenge of automation in control systems posed by ICE aircraft design and objectives, introduce the research objective of this project as **designing an online model-free adaptive controller for the over-actuated and highly maneuverable ICE aircraft.** In summary, the focus of this study is the optimal control of a single controller in interaction with the environment in the context of autonomous flight.

Problem: The control problem of the ICE model deals with two main challenges in symmetric and asymmetric flights: the optimal control allocation problem and the reference tracking problem. Also, the lack of global knowledge about the complex nonlinear system dynamics attenuates the performance of the model-based approaches in online adaptation. Therefore, an online and self-learning control approach should be taken to deal with uncertainties and almighty faults during aircraft mission and toward the completion of its flight envelope. The current over-actuated ICE model is highly nonlinear and complex, caused by the inherently unstable design of the aircraft and its over-actuated configuration. These challenges confirm the requirement for the design of a controller, which is independent of the system model. The over-actuation feature demands a control allocation approach that handles nonlinearities. One of the limitations with the developed controllers for the ICE aircraft is their dependence on a model of the system.

Methods: The model-free nature of the reinforcement learning methods, which leads to the inherent handling of the input constraints, makes them a suitable candidate for the ICE automated control system. We propose continuous RL methods, which include the control allocation problem and perform the control task with no prior knowledge about the system and eliminate the introduction of an additional control allocation method. Also, in a recent study that investigates the application of discrete RL method of Q-learning for the ICE model, it was found that the performance of the discrete approach is less than that of the model-based methods in fulfilling the control objective completely. These limitations can overcome with the application of continuous RL methods for the ICE aircraft. The continuous RL utilizes the approximation power for the identification of value functions (policy evaluation) and policy improvements. Following this step, the optimal policy which corresponds to the optimal control input will be selected. Hence, this study first examines different RL approaches and methods followed by choice of the suitable continuous process of Heuristic Dynamic Programming (HDP), an extension of Adaptive Critic Designs (ACD) for the design of the controller.

Case Studies: To become familiar with the configuration and control concepts of the ICE aircraft, the design objectives are studied. Further, the available SIMULINK and Spline aerodynamic models of the aircraft are examined. The purpose is to explore the possible experiment and simulation grounds for the implementation of the RL methods. As part of the preliminary study and to practice the application of the discrete and continuous RL methods, a simple 2-state swing-up pendulum problem and the nonlinear missile tracking problem are introduced. The two approaches of discrete Q-learning and continuous Heuristic Dynamic Programming (HDP), which is an actor-critic architecture are investigated for these case studies, respectively, followed by the evaluation of the results. Next, the application of the approximate RL controller is performed for the control problem of the ICE aircraft in two objectives of the angle of attack tracking and altitude tracking.

Conclusion and Next Steps: The primary focus of this MSc thesis project is on continuous RL control methods for the ICE aircraft. For this purpose, this study explores suitable generalization methods, the

available ICE configuration, and representative flight conditions to evaluate the online performances of the controller. To investigate the adaptability of the RL controller, the controller task is defined for the aircraft in achieving an abstract objective in specified flight conditions and during changes to the tracking signal. The results are compared with other RL methods found in the literature about the ICE control problem. The proposed design is shown to enhance the control performance significantly compared to discrete RL-based. It also has the advantage of quick online learning and adaptation while being robust.

Contribution: Presenting the capabilities of continuous RL methods in aerospace applications contributes to automation research for flight control systems related to innovative designs. The use of these methods for ICE aircraft provides the RL research community with an implementation of the RL methods for an innovative over-actuated system and gives novel insights to the developers of the ICE automatic control systems.

Preface

This document contains the main results for the MSc graduation project at the Control and Simulation profile, Control and Operation track, faculty of Aerospace Engineering at the TU Delft. The general objective of this thesis project is to study the competency of one of the most promising Machine Learning (ML) techniques, i.e., Reinforcement Learning (RL), in online solving of the nonlinear control problems for unknown systems such as Innovative Control Effector (ICE) aircraft. Further to investigate the ability of the controller to adapt to changes in the control problem as a step towards automation.

Given that the over-actuated autonomous aircraft are the next generation of fighters and considering the previous limitations with discrete RL-based controllers, the topic of this thesis is established as **"The application of Continuous Reinforcement Learning methods on Innovative Control Effector (ICE) aircraft model"**. This report demonstrates the knowledge developed by the student on the proposed topic and the results of the implementation.

This thesis is written mostly from an academic perspective, using publicly available information. The focus is on the theoretical analysis of the implementation of a continuous RL-based controller and simulation experiments on the ICE aircraft model, considering the over-actuation and non-linearity features in the system. The results will provide educational benefits for the RL community as well as for researchers in the field of aerospace automation. This study, however, does not address the practical application of the results as well as possible industry perspectives. However, this does not result in avoiding the benefits of RL methods to be seen as potential solutions for the ICE control challenge.

Acknowledgement

I would like to thank my daily supervisor Dr. E. van Kampen who made this master thesis possible for me and for making the time available to share his knowledge and point me in the right direction. I appreciate his patience and understanding that helped me to shape this work from beginning to end. I would also wish to thank Dr. Chu for our important meetings and Dr. C.C. de Visser for our discussions and his feedbacks on the ICE aircraft model.

I want to thank my father, Behnam, and my mother, Maryam, for their unconditional support and their hard work. I also want to thank my brother, Ardalan, for his help and positive energy.

In the end, I hope that my effort in graduating from the Aerospace Engineering department of Delft University will motivate other female engineers in my country and worldwide to believe in themselves and follow the path of their interest.

Contents

List of Figures

List of Tables

List of symbols

a	Action
A	Action space
a_z	Acceleration along the aircraft's Z-axis
b	Aerodynamic coefficients or wing span
c	Immediate cost value
c_α	Learning rates update coefficient
$\bar{c}$	Mean aerodynamic chord
C	Cost function constant
C_z, C_m	Longitudinal force and pitch moment coefficients
d_{TV}	Thrust vectoring arm
d_l	Reference length
E	Expectation or error
f	Transition probability density function
F_x, F_y, F_z	Aerodynamic body forces
g	Gravitational acceleration
h	Missile aerodynamic coefficients
I_{yy}	Pitching moment of inertia
J	Value or cost-to-go function
l, m, n	Aerodynamic moments in body frame
M	Mach number
M_x, M_y, M_z	Aerodynamic moments
m_a	Missile mass
n	Number of hidden layer neurons
N	Number of inputs to the network
$p(x^{'}\|x, a)$	Probability of transition to state x', from state$x$ with action a
p, q, r	Pitch, roll and yaw angular rates
$\bar{q}$	Dynamic pressure
Q	Q-function or state-action value function
Q_c	Cost function state weight matrix
Q_m	Model error weight
r	Immediate reward value
R	Cumulative reward
R	Cost function control input weight matrix
s	State value
S	Wing surface
S	State space
t	Time
T	Thrust force vector or torque
u, v, w	Airspeed components
u	Control input vector
U	Utility function
V	True speed
$V(x)$	Value function
V_T	Speed
W	Neural networks weights
W_0	Initial weights

W^h, W^o	Hidden and output layer weights
W_c, W_a	Critic and actor networks weights
x	State vector
$x^{'}$	Next state value, interchangeable with $x(t+1)$

Greek Symbols

α	Angle of attack
π	Policy or control law
$\pi(a\|s)$	Probability of taking action$a$ in state s under stochastic policy pi
γ	Discount factor
α	Learning rate
α	Angle of attack
θ	Angular position
$\hat{\theta}$	Policy function approximation parameter
$\dot{\theta}$	Angular velocity
μ	Friction
$\sum$	Sum
ϵ	Probability of choosing an action with maximum long-term reward in greedy policy
ϵ	Neural networks approximation error
λ	Eligibility trace
∇	Partial derivative
ϕ	Activation function
δ_e	Elevator deflection
σ	Noise amplitude

1

Introduction

This chapter includes the introduction for the research on the application of continuous reinforcement learning for the Innovative Control Effector (ICE) aircraft. The chapter is structured as follows. First, the motivation and research context are presented in section 1.1. In section 1.2, an outline of the previous studies is provided, followed by the problem statement. Finally, the research objective and questions are formulated in section 1.3. section 1.4 subsequently presents the central thesis report outline..

1.1. Motivation and Research Context

Higher levels of autonomy in aerospace systems is an urgent requirement, considering the growing system complexities, the increase in control task difficulties, and therefore the need for adaptability (Zhou, 2018). The aim of automation in aircraft control systems calls for innovative approaches that are only possible with a revolution in the Flight Control System (FCS) designs. As a step toward automation, the automated agent should be equipped with adaptability qualities. A genuinely autonomous control system can adjust its behaviour after unexpected changes occurred in the system. In the case of a combat aircraft, examples of unknown and dangerous situations can be combat damage affecting control surfaces or asymmetrical weight distribution after missed/faulty weapons release. For this reason, the aspect of self-learning and adaptability in unexpected conditions is one of the most critical challenges in control systems that are designed for automation.

Technological improvements in aerospace systems have spurred innovative designs both in the civilian and in the military sectors. One of these designs is the Innovative Control Effector (ICE), a wedge-shaped aircraft with no vertical surfaces (either fixed or moving) conceived to combine the simplicity of its shape with complex FCS to achieve very low radar signature. Figure 1.1 shows the wing-shape configuration of the aircraft, without the control suits. The ICE aircraft designed control objectives are particularly based upon maneuverability, stealth, and resilience that are the basic attributes of modern-day fighter combat aircraft (Dorsett and Mehl (1996), Bowlus *et al.* (1997), Niestroy *et al.* (2017) and Whitford (1987)). A large number of highly non-linear and constantly interacting control effectors in tail-less ICE configuration, while satisfying the design requirements of the modern fighters, introduce a challenging control problem. A continuous controller is required that ensures the online, autonomous and contiguous intervention of FCS, particularly for failures (Thrun and Littman, 2000). The desire controller should use the combination of the control effectors in their control limits, to avoid extensive drag and effector utilization in achieving the optimal control objective. Also, the control system should be suitable for adaptability, independent of the global model of the system. Therefore, quick online and model-free adaptation to the changes is required.

Reinforcement Learning (RL), which is the idea of active decision-making and learning by interacting with the environment, is relatively new to the field of Guidance, Navigation and Control (GNC) of aerospace systems. Still, RL solutions have shown promising results in model-free adaptive optimal

control problems for both discrete and continuous systems (Zhou *et al.* (2015), Zhou *et al.* (2017), Zhou *et al.* (2018), de Vries (2017), Ferrari and Stengel (2004) and Ferrari *et al.* (2008)). Adaptive optimal controllers are usually designed to find optimal solutions for user-defined performance equations. They use system identification methods to find the local model parameters and then use the model to solve the optimal equations. Instead, RL controllers are introduced for adaptive control of the real-world complex systems, when a model of the system is generally unavailable, and therefore, model-based approaches become invalid. The online adaptability is achieved by being independent of the global model updates, possible by goal-oriented and incremental characteristics of RL. Instead of instructing the agent on how to do things from the model, the agent comes up with the suitable control command using the end-goal alone. RL can also be seen as an optimal control approach. The current optimal control methods frequently require complete knowledge of the system and are normally designed to solve an optimization problem offline, e.g., by solving Hamilton-Jacobi-Bellman (HJB) equations. Unlike these controllers, the recently developed RL optimization methods (e.g., based on Adaptive Critic Designs (ACDs) architecture) allows the design of controllers that solve optimal equations independent of the model of the system dynamics (Lewis *et al.*, 2012).

RL methods are studied for discrete and continuous dynamical systems. The use of the tabular RL methods for control of the continuous systems exhibits low-performance characteristics when compared to current model-based control approaches (de Vries, 2017). Introducing approximation methods for these methods overcome the deficiencies of the discrete RL, keeping model-free characteristic and adding online self-learning adaptability feature. Hence, approximate RL is conceived as a suitable approach for the development of an automated FCS as discussed by Ferrari and Stengel (2004), Ferrari *et al.* (2008) and Zhou (2018). The offline learned RL agent with the optimal policy can be utilized for online and automated adaptability of the aircraft to, e.g., unknown dynamics changes. Recently, the performance of continuous RL methods has improved with the extension of optimally for RL methods of Approximate Dynamic Programming (ADP) such as Neuro-Dynamic Programming (NDP) and Spline-based Dynamic Programming which differ in their approach for approximation and can be used in an ACD framework to provide a powerful RL control architecture(Powell (2009), Bradtke and Barto (1996) and Eerland *et al.* (2016)). Therefore, approximate or continuous RL is a promising approach for adaptive and optimal control of complex systems. In this thesis, the implementation of approximate RL for an over-actuated system of the ICE aircraft will be discussed.

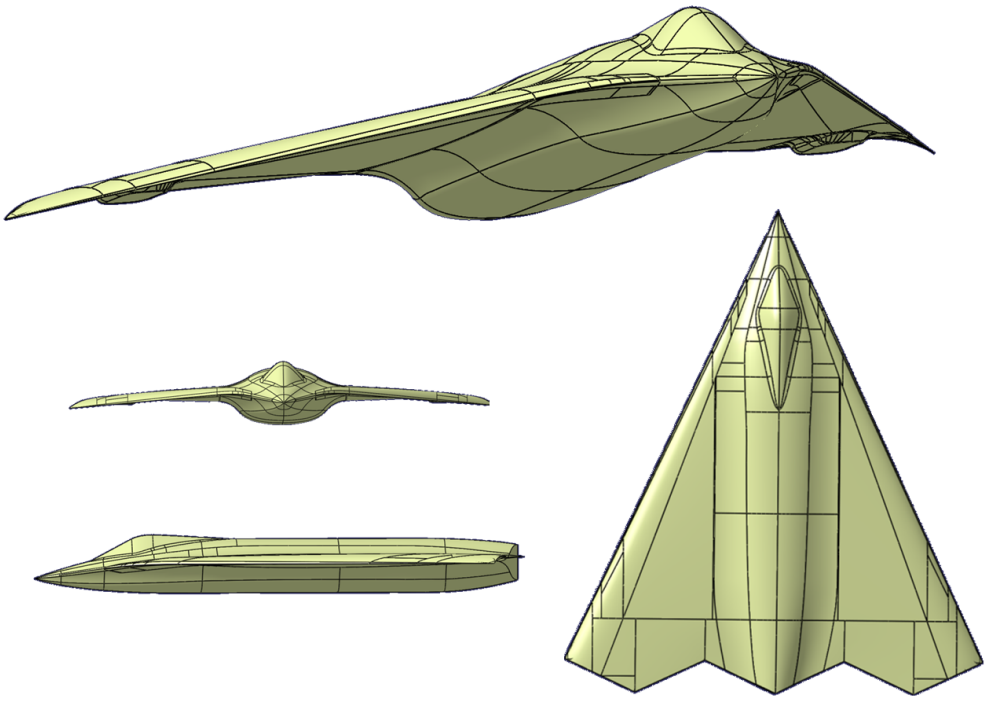


Figure (1.1) The Innovative Control Effector (ICE) research aircraft

1.2. Problem Statement

In this section, first, the ideal FCS for current and future autonomous ICE models is described, followed by a background on the current autonomous FCSs. Later, it is explained why it is critical to change the existing systems and what are the benefits of such a change. Finally, the RL solution is introduced, and the thesis statement is presented.

The study on the development of FCSs is ongoing research. The Advance or Automatic Flight Control Systems (AFCS) employ the aerodynamic control surfaces to handle the moments and forces on the aircraft to control its slow dynamics (Lombaerts, 2011). The goal is to do this process with the least assistant from the human operator while satisfying the handling qualities requirements of a military aircraft. The performance of an AFCS is highly dependent on the control laws used for translating sensor measurements to control surface outputs, i.e., the control allocation techniques. For a redundant actuation aircraft, the number of control inputs is larger than the number of control variables, and hence control allocation is required to attain a control command that would result in optimal control of the aircraft.

In a recent study at the Delft University of Technology, an Incremental Nonlinear Control Allocation (INCA) method was applied to the ICE model. Although this approach presents promising results for tracking and control allocation performance of the ICE aircraft in real-time, it assumes full state feedback in the control loop and is, therefore, a model-based method (Matamoros Cid, 2017).

In another recent literature, a discrete RL-based control allocation method is developed for the ICE model to fulfil an objective of altitude control (de Vries, 2017). In this study, it is found that the performance of the discrete RL is less than that of model-based control methods. This shortcoming is explained to be due to the limitations in the function approximations. The same study proposes a hybrid approach of combining model-bounded control allocation with RL-based methods with the former to be used in normal flight conditions and the latter in unforeseen circumstances.

van Kampen *et al.* (2006) focused on the self-learning and the adaptability of RL methods and their advantages in the field of re-configurable flight control, where accidental changes in the plant dynamics require control system adaptation. He describes the prominence of RL comparing to supervised learning in being free of prior instructions for achieving a specific high-level goal and its ability to come up with policies and actions without restrictions of a structured model. Rather than instructing the controller with necessary control actions, which might not be available on every time instances, an RL controller is given the desired states and should learn the optimal control actions for itself. By comparing two continuous actor/critic RL methods, he concluded that the continuous domain brings a broader scope of applications for RL, comparing to the discrete algorithms.

A continuous reinforcement learning method is a promising approach to overcome the limitations of the discrete reinforcement learning controller and make the online, model-free learning and the inherent control allocation possible. These are the main requirements for the development of an automated FCS. It should be noted that by continuous, it is meant here that a system which is continuous in states and actions but not necessarily in time (e.g., discrete-time and continuous-time systems). Although real systems are continuous in time, however, their measurements are obtained discrete. The offline learned RL agent with the optimal policy can be utilized for online and automated adaptation (robustness of the controller) of the aircraft in an unknown environment, or the same agent can learn the optimal policy with no prior offline training using more advanced architectures. Further, the convergence with the extension of optimally for RL methods of Approximate Dynamic Programming ADP in the continuous space, is assured in literature by Powell (2009), Bradtke and Barto (1996) and Eerland *et al.* (2016). This makes the continuous RL algorithms a suitable choice for designing an online adaptive and optimal controller. Therefore, the problem statement for this thesis is defined as:

"The study of an online, optimal, and model-free adaptive controller to be implemented for the over-actuated autonomous ICE aircraft model, enabling the achievement of longitudinal control objectives in a pre-defined flight condition."

1.3. Research Objectives and Question

Looking at the motivation and research context, it can be seen that designing a controller based on approximate RL methods can improve automation in FCS, particularly in situations where the agent is dealing with unforeseen events. These situations can be resolved by online, model-free, and adaptive characteristics of RL solutions. These features, exclusively, bring significant benefits for the next generation of fighters aiming for automated technologies. Exploiting RL algorithms for solving control problems handles the control allocation problem inherently for a given objective. Following the above research problem, the main aim of the thesis can be formulated as:

Research Objective: *"Adaptive and Online control of an over-actuated control suite aircraft such as the ICE by development and implementation of an approximate reinforcement learning-based robust and adaptive controller, with no prior knowledge of the system."*

To have a basis for measuring the achievement of the research goals by the completion of this thesis work, the following hypothesis is adjusted:

Hypothesis: *"A continuous adaptive and model-free RL-based controller can be developed which can optimally control the unknown over-actuated ICE aircraft in routine flight conditions and in case of changes in the control objective."*

Based on the information presented for research goals and hypotheses, the knowledge required to achieve the project aims is identified. In the following list, these requirements are reflected as research questions and sub-questions associated with them:

1. What is the state-of-the-art in the field of Adaptive and Robust control with Reinforcement Learning?
    - What are the recent achievements in the subject of approximate reinforcement learning?
    - What are the latest developments in approximate dynamic programming methods in the context of aerospace systems?
    - What are the latest developments in actor-critic design methods?
    - What are the best function approximation methods for online and large states applications, in terms of data efficiency and computational costs?
2. Which control task is suitable for the ICE model to present the online adaptability and performance of the RL methods?
    - What are the objectives of the ICE design, in terms of its mission?
    - What requirements these objectives would imply from the control perspective?
    - What are the pros and cons of the methods used for the control problem of the over-actuated ICE model so far?
3. How can the performance of an approximate RL-based control system be compared to the Discrete RL-based controller?
    - Is the tracking error less for the approximate RL-based controller?
    - Is the designed controller faster in achieving the control objective?
4. How is the performance of the controller designed by continuous RL methods influenced by the hyper-parameters?
    - How function approximation parameters affect the RL controller?
    - How learning parameters affect the RL controller?
5. How adaptive and robust is the approximate RL-based FCS to changes?
    - How robust is the online learned controller to change in the reference signal?

- How adaptive is the controller to changes in the reference signal?

The first two research questions and part of the third research question are answered by the preliminary phase of the thesis project so the literature study phase. The last two questions, including the more comprehensive answer to the third question, are then answered by the second part of the thesis study which concludes this thesis project.

1.4. Report Outline

In this document, a research framework is presented, which includes the elaboration of the following topics. The introduction of the RL and the state-of-the-art findings in RL-based control is shown in chapter 2. A background on adaptive and optimal flight controllers and the location of RL in these concepts are given in chapter 3. The ICE model and configuration in addition to its control challenges are described in chapter 4. chapter 5 includes the simulation set-up and RL control experiments for the swing-up pendulum problem and missile nonlinear tracking control problem. The results for the application of the approximate reinforcement learning controller for the ICE aircraft model are presented by the scientific paper. A sensitivity analysis is performed on the controller hyper-parameters in chapter 6, followed by the robustness and adaptability check of the designed controller. Finally, the conclusion of this thesis project and the next steps for future studies are given by chapter 7. Figure 2.1 shows the roadmap and the orders in which the chapters of this report can be read.

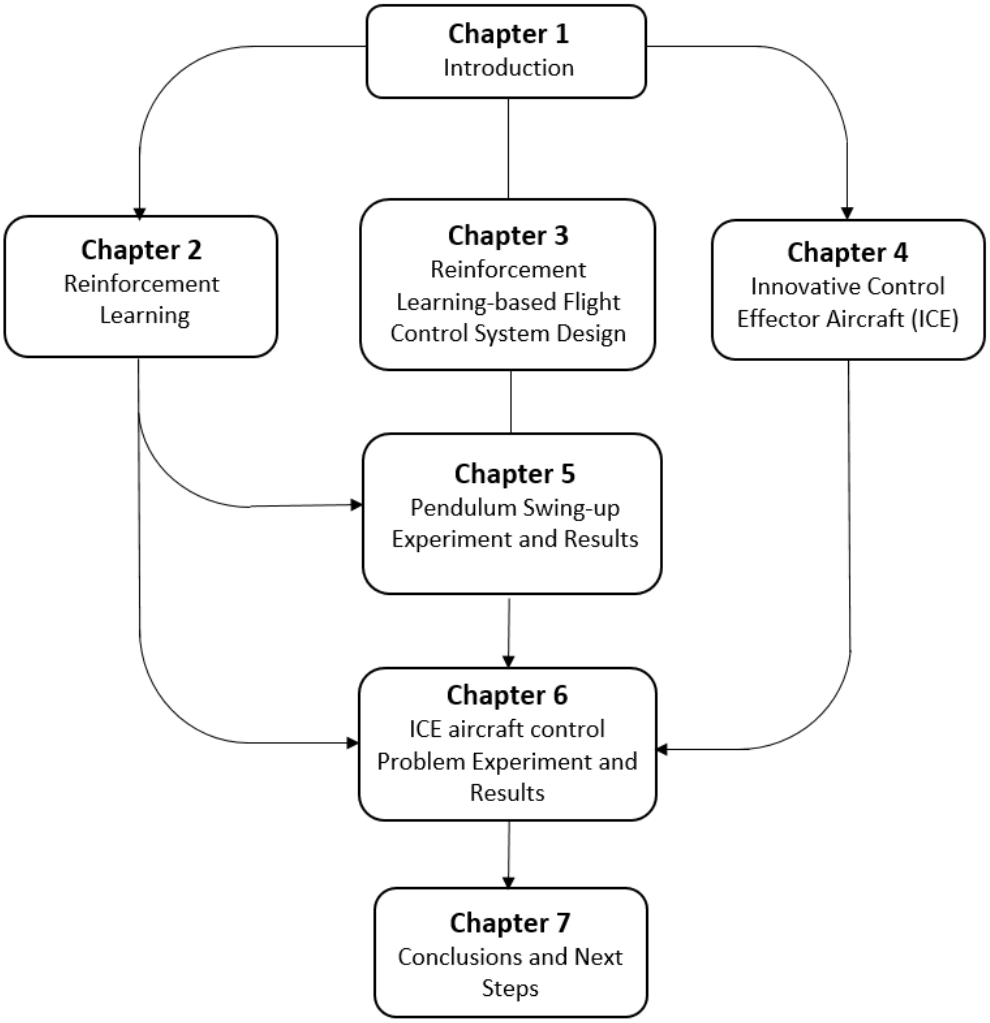


Figure (1.2) Schematic road map of the thesis report

I

Conference Paper

Online Actor-Critic-Based Adaptive Control for a Tailless Aircraft with Innovative Control Effectors

K. Shayan*
Delft University of Technology, 2629HS Delft, The Netherlands.

Higher levels of autonomy in aerospace systems is an urgent requirement, considering the increase in control task difficulties, and the need for adaptability of the complex systems. Reinforcement learning (RL) control is one of the promising approaches for adaptive control of air vehicles that are designed for automation. Conventional discrete reinforcement learning methods fail in providing satisfactory performance for flight control systems (FCSs), especially for a complex configuration of a tailless over-actuated aircraft. The lack of efficiency of the discrete controller in exploration for finding the optimal policy, the so-called problem of 'curse of dimensionality', results in an approach that is not suitable for online implementation. Also, the achieved discrete non-smooth control policy usually does not apply to the real world control surfaces. This paper studies the experiments with Heuristic Dynamic Programming (HDP), a method obtained from adaptive critic design (ACDs), as a continuous reinforcement learning approach. ACD methods can capture the nonlinearities in the complex dynamics of the aircraft while solving the control problem computationally efficient by using continuous states and action spaces. Such qualities make ACDs suitable for online FCS design for unstable systems like tailless aircraft. In this paper, the ACD-based controller is developed and implemented for the Innovative Control Effector (ICE) aircraft, a highly maneuverable aircraft with redundancy in its control effectors suite. The coupled control effectors configuration has strong interactions and, therefore, proposes a need for proper control allocation. The online simulation results show the accuracy of the designed continuous RL controller in the longitudinal control of the aircraft using different sets of control effectors. The proposed approach also shows significant improvements in the tracking performance and control policy smoothness (e.g., compared to discrete methods).

Nomenclature

b	=	Wing span
c	=	Immediate cost value
c_α	=	Learning rates update coefficient
$\bar{c}$	=	Mean aerodynamic chord
C	=	Cost function constant
C_z, C_m	=	Longitudinal force and pitch moment coefficients
d_{TV}	=	Thrust vectoring arm
E	=	Error
F_x, F_y, F_z	=	Aerodynamic body forces
g	=	Gravitational acceleration
I_{yy}	=	Pitching moment of inertia
J	=	Value or cost-to-go function
l, m, n	=	Aerodynamic moments in body frame
M	=	Mach number
M_x, M_y, M_z	=	Aerodynamic moments
n	=	Number of hidden layer neurons

*MSc Student, Control and Simulation Division, Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delf, The Netherlands, k.shayan@student.tudelft.nl. AIAA Member.

N	=	Number of inputs to the network
p, q, r	=	Pitch, roll and yaw angular rates
$\bar{q}$	=	Dynamic pressure
Q_c	=	Cost function state weight matrix
Q_m	=	Model error weight
$\boldsymbol{r}$	=	Immediate reward value
R	=	Cumulative reward
$\boldsymbol{R}$	=	Cost function control input weight matrix
S	=	Wing surface
t	=	Time
T	=	Thrust force vector
u, v, w	=	Airspeed components
$\boldsymbol{u}$	=	Control input vector
V	=	True speed
$V(x)$	=	Value function
W	=	Neural networks weights
W_0	=	Initial weights
W^h, W^o	=	Hidden and output layer weights
W_c, W_a	=	Critic and actor networks weights
x	=	State vector

Subscripts

$\boldsymbol{\alpha}$	=	Angle of attack
$\boldsymbol{\alpha}$	=	Learning rate
δ	=	Control effectors deflection
γ	=	Discount factor
$\pi(s)$	=	Policy function
$\boldsymbol{\phi}, \theta, \psi$	=	Roll, pitch and yaw angles
$\boldsymbol{\phi}, \boldsymbol{\sigma}$	=	Activation Function
ρ	=	Atmospheric pressure

I. Introduction

The aim for automation in aircraft control systems calls for innovative approaches that are only possible with a revolution in the current Flight Control System (FCS) designs. Based on a recent study, loss of control (LOC) is the main reason for jet aircraft fatalities [1]. A genuinely autonomous control system can adjust its behavior after unexpected changes occurred in the system, so show adaptability features. With a combat aircraft, examples of unknown and dangerous situations can be combat damage affecting control surfaces or asymmetrical weight distribution after missed/faulty weapons release. As a result, self-learning and adaptability in unknown conditions are one of the most critical challenges in control systems designed for automation. Adaptive control is one of the most promising approaches in automatic control for the prevention of LOC. Between adaptive control methods, reinforcement learning due to its learning abilities attracts many researchers in this field.

Technological improvements in aerospace systems have spurred innovative designs both in the civilian and in the military sectors. The innovative control effector aircraft (ICE) is state-of-the-art in the design of the future fighter aircraft. The low-observability that resulted from the tailless configuration and the high maneuverability reflected in its unconventional control suite make ICE aircraft unique in its design. Maneuverability, stealth, and resilience are the primary attributes of modern-day fighter combat aircraft that also provoke the ICE aircraft design control objectives ([2–5]). A large number of highly non-linear and constantly interacting control effectors in tail-less ICE configuration, while satisfying the design requirements of the modern fighters, introduce a challenging control problem. The need for a continuous controller that ensures the online, autonomous, and contiguous intervention of FCS (particularly for failures) is then necessary [6]. The desire controller should use the combination of the control effectors in their control limits, to

avoid extensive drag and effector utilization in achieving the optimal control objective. Also, the control system should be suitable for adaptability, independent of the global model of the system. Therefore, quick online adaptation to the possible changes is required.

Reinforcement Learning (RL) is the idea of active decision-making and learning by interacting with the environment. RL is relatively new to the field of Guidance, Navigation, and Control (GNC) of aerospace systems. Still, RL solutions have shown promising results in model-free adaptive optimal control problems for both discrete and continuous systems [7–12]. Adaptive optimal controllers are usually designed to find optimal solutions for user-defined performance equations. They use system identification methods to find the global model parameters and then use the model to solve optimal equations. Instead, RL controllers are introduced for adaptive control of the real-world complex systems, when a model of the system is generally unavailable, and therefore, model-based approaches become invalid. The online adaptability is achieved by being independent of the global model updates, possible by goal-oriented characteristics of RL. Instead of instructing the agent on how to do things from the model, the agent comes up with the suitable control command using the end-goal alone. RL can also be seen as an optimal control approach. The current optimal control methods frequently require complete knowledge of the system and are generally designed to solve an optimization problem offline, e.g., by solving Hamilton-Jacobi-Bellman (HJB) equations. Unlike these controllers, the recently developed RL optimization methods allows the design of controllers that solve optimal equations independent of the model of the system dynamics [13].

RL methods are studied for discrete and continuous dynamical systems. Using the tabular RL methods for control of the continuous systems exhibits low-performance characteristics, when compared to current model-based control approaches [10]. Introducing approximation methods for these methods overcome the deficiencies of the discrete RL while keeping the model-free, and self-learning adaptability characteristics. Hence, approximate RL is conceived as a suitable approach for the development of an automated FCS, as discussed by [11, 12, 14]. The offline learned RL agent with the optimal policy can be used for online and automatic adaptability of the aircraft to, e.g., unknown dynamics changes. Recently, the performance of continuous RL methods has improved with the extension of optimally for RL methods of Adaptive Critic Designs (ACDs) categorized within the Actor-Critic (AC) framework. The technique integrates dynamic programming, temporal-difference (TD) learning, and function approximation to provide model-free and online learning in the continuous domain.

A discrete RL-based controller using the Q-learning approach was developed for the ICE model to fulfill an objective of altitude control [10]. The study shows that the performance of the discrete RL is less than that of model-based approaches. The limitations that come with the discretization of the state and action space result in the discrete RL controllers deficiency. Van Kampen et al. [15] focused on the self-learning and the adaptability of continuous RL methods. He showed their advantages in the field of re-configurable flight control, where accidental changes in the plant dynamics require the control system adaptation. He describes the prominence of RL comparing to supervised learning in being free of prior instructions for achieving a specific goal and its ability to come up with policies without restrictions of a structured model. By comparing two continuous ACD methods of HDP and Action-Dependent HDP (ADHDP), he concluded that the continuous domain brings a broader scope of applications for RL (compared to the discrete algorithms). The same study shows the high performance of the HDP approach in terms of success ratio and adaptation speed (compared to action-dependent approach).

This paper propose the implementation of one of the promising approaches of adaptive critic designs, the so-called heuristic dynamic programming method, to overcome the deficiencies of the discrete reinforcement learning control. Online, model-free, and self-learning are the main requirements for the development of an automated FCS that can be achieved by an RL controller. By continuous, it is meant here that the system is continuous in states and actions but not necessarily in time (e.g., discrete-time and continuous-time systems). Although real systems are continuous in time, however, their measurements are obtained discrete. With the advance HDP architecture, the RL controller can achieve the (near-) optimal policy with no prior knowledge about the complex dynamics of a system such as ICE aircraft. The convergence with the extension of optimally for RL methods in a continuous space is assured in literature by [16–18]. This makes the HDP architecture a suitable choice for designing an online, model-free, and adaptive controller for the ICE aircraft. In the following, by all effectors it is meant that all 12 effectors of the ICE aircraft, except for the thrust vectoring.

II. The Actor-Critic Reinforcement Learning Problem

In most of the reinforcement learning problems, a state is measured, and an action corresponding to that state is chosen. Next, a transition to the next state is measured, and the reward/cost value restores for such a shift. The long-term reward/cost values are called value function. Such a formulation of the problem is based on the Markov Decision Process (MDP) modeling. The framework includes two processes of policy evaluation (value function improvement) and policy improvement (policy function update) to reach the optimal policy. Fig.1 shows a schematic of such an interaction. RL, in the context of control, involves the change of the controller's strategy based on its experience to maximize long-term rewards or minimize a cost function. In case of constraints for the control limits, the designer defines the boundaries for the learning controller. To reach optimality in an RL framework, the importance of exploration and its trade-off with exploitation should not be ignored. The smart design of the reward or cost function also plays a vital role in convergence to the correct optimal policy.

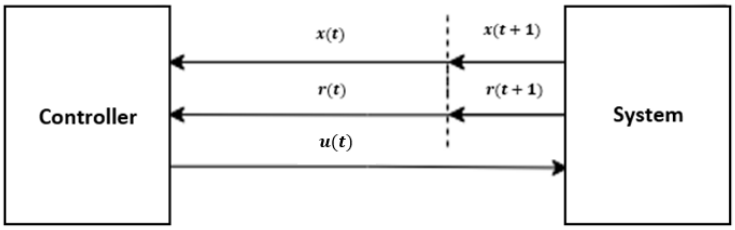


Fig. 1 A schematic diagram of the controller and system interaction in a RL problem

Within continuous RL approaches, AC methods are emphasized more for non-stationary settings that can deal with continuous state and action spaces. AC combines learning in policy space with simultaneous value function approximations that result in algorithms that are proved for convergence. An AC framework integrates three fundamental aspects and classes of RL, Dynamic Programming (DP), Temporal-Difference (TD) learning, and Function Approximation (FA). By combining the DP method with incremental TD approach, possible within approximate solutions of FAs, the AC design can overcome the deficiencies of each technique and provide an architecture suitable for online and model-free learning of large-scale processes. This structure also has shown promising results for optimal adaptive control of non-linear systems by exploiting a variety of sophisticated FA methods. In the following this method is described in more details.

A. Actor-Critic Architecture

Actor-critic methods are a group of continuous RL approaches, in which classification is done if the critic element is approximating the expected total future reward/cost or the value function, and/or the derivative of the value function. Such a scheme abandoned the knowledge of the dynamics, by approximation of the Hamilton-Jacobian-Bellman (HJB) equation. The state information obtained by the critic will be used for determining the value of the state and change the internal parameters of the critic so that this knowledge will be stored compactly within the critic parameters. The actor represents the policy π of the controller by providing actions based on the current information of the state. The optimal policy is achieved then by making the approximated value function $\hat{J}(t)$ to a goal value $J^*(t)$ using the actor training error. Fig.2 represents the schematic of the AC framework.

AC methods of approximation can be seen as an extension of Generalized Policy Iteration (GPI) [19]) to the continuous state and action spaces. In a GPI framework, the critic policy evaluation step is not performed entirely, and the current estimation of the value function would be updated towards the value of the policy [20]. However, in an actor-critic structure, both critic and actor updates would be done completely and simultaneously at each time step. Since the TD error is used as the update rule at each time step, it is not needed to wait for the end of each episode, which makes the AC structures suitable for online application. Also, no greedy selection (so selecting based on an optimization process) is performed, and it is the policy itself that determines the direction.

Within AC methods, HDP algorithms are one of the most used and popular architectures. Their simplicity in implementation while showing high performance in adaptation and convergence rate besides their high success ratio (for example, when compared with other AC methods such as ADHDP methods [15]) makes them a suitable choice for

adaptive control of complex nonlinear systems. In the below a more detail of this framework is described.

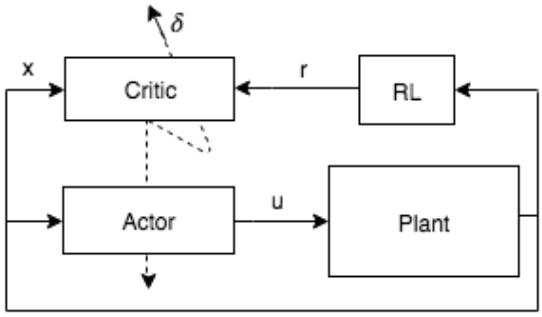


Fig. 2 Schematic of the actor-critic structure

B. Heuristic Dynamic Programming

In a HDP framework, the critic uses only the incoming state information for value-function approximation, while the actor network is providing the control input. There should be an information exchange between the two networks. Since the actor needs the information updates in the critic as its output indirectly influences also the critic output. Therefore, in such cases, an internal model of the plant is approximated and will be updated along with the critic and actor networks to identify and reflect the changes in the real plant [21]. The optimal value function is usually unknown and, therefore, cannot be used as the target value for the learning, although the recursive characteristics of it help the network move toward the optimal value function. A schematic of an HDP architecture is given in Fig.3.

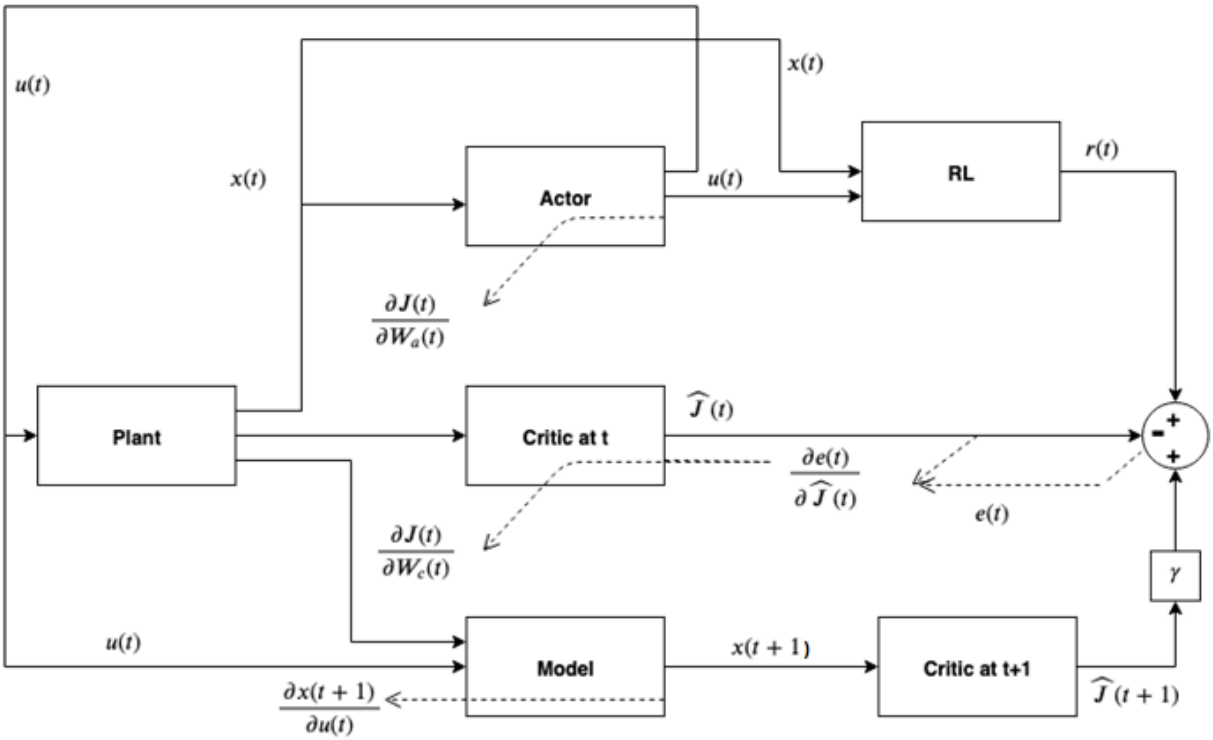


Fig. 3 HDP scheme block diagrams and updates based on [19]

For HDP, an indirect path from the actor output to the approximated plant dynamics and then through the critic will approximate the value-function, and through the same path, the actor network will be updated in the direction to achieve the zero value for actor error $E_a(t)$ [15]. While the critic is used for approximation of the value function, the unbalance in the Bellman's equation will be used as the learning error for the actor that will need to be minimized. The actor would then lead the system to the regions within the state space that the value function approximation of the critic has high or low values depending on the way the reward or cost function is defined.

It is important to note that the only place where the plant dynamics will be used is in the back-propagation step for the actor network. Also, in such an approach, there is no direct connection between the actor output to the critic network, and therefore, the critic is estimating the value-function now only based on the state information. Thus, the task of the critic is simplified, and the complexity is now partially moved to the plant dynamics. In the HDP framework, the plant network is independent, and it would influence the actor and critic networks. In the following, the actor, critic and model networks are defined for the HDP framework with this assumption that both networks are approximated with two-layer Convolutional Artificial Neural networks (CANNs) with one hidden layer. It is assumed that the networks are updated using the gradient descent method, and a general input-output definition is given that will be used later in this paper for the development of the controller.

1. Critic Network

The input to the critic network is the difference between the measured state $x(t)$ and the reference value $x_{ref}(t)$ so the tracking error $e(t)$ defined by Eq. (1).

$$e(t) = x(t) - x_{ref}(t) \tag{1}$$

The output is the estimated value function $\widehat{J}(X(t))$. The network is defined then by:

$$\begin{aligned} \widehat{J}(t) &= W_c^o(t)\boldsymbol{\phi}(W_c^h e(t)) \\ &= \sum_{i=1}^{n_c} \widehat{w}_{ci}^o(t)\boldsymbol{\phi}_i(\sum_{j=1}^{N} \widehat{w}_{cij}^h(t)e_j(t)) \end{aligned} \tag{2}$$

where $\boldsymbol{\phi}(t) = [\boldsymbol{\phi}_1(t), ..., \boldsymbol{\phi}_{n_c}(t)]^T \in R^{n_c}$ is the hidden layer activation function vector (e.g. tanh), W_c^h is the output layer weights defined by as:

$$W_c^h = \begin{bmatrix} w_{c11}^h & \cdots & w_{c1(N)}^h \\ \vdots & \vdots & \vdots \\ w_{cn_c1}^h & \cdots & w_{cn_c(N)}^h \end{bmatrix} \in R^{n_c \times N} \tag{3}$$

where n_c is the number of the neurons for the hidden layer and w_{cij}^h denotes the weight from the jth input value to the ith hidden neuron. The output layer weights are defined as:

$$W_c^o = [w_{c1}^o, w_{c2}^o, ..., w_{cn_c}^o] \in R^{1 \times n_c} \tag{4}$$

where w_{ci}^o is the weight from ith hidden layer to the output layer.

2. Actor Network

This NN approximates the control input for the nonlinear tracking problem. The data to the actor network is the same as the critic, and the output is the control command given as:

$$\widehat{u}(t, W_a^h, W_a^o) = [\widehat{u}_1, ..., \widehat{u}_m] \tag{5}$$

where

$$\begin{aligned}\widehat{u}(t) &= W_a^o(t)\boldsymbol{\sigma}(W_a^h(t)e(t)) \\ &= \sum_{k=1}^{n_a} \widehat{w}_{a_k}^o(t)\boldsymbol{\sigma}_k(\sum_{l=1}^{N} \widehat{w}_{a_{kl}}^h(t)e_l(t))\end{aligned} \tag{6}$$

where $\boldsymbol{\sigma}(t) = [\boldsymbol{\sigma}_1(t), ..., \boldsymbol{\sigma}_{n_a}(t)]^T \in R^{n_a}$ is the hidden layer activation function vector, n_a is the number of neurons for the hidden layer and $k = 1, ..., m$ is the kth output-layer. The weights matrix for hidden layer is defined as:

$$W_a^h = \begin{bmatrix} w_{a11}^h & \cdots & w_{a1(N)}^h \\ \vdots & \vdots & \vdots \\ w_{an_a1}^h & \cdots & w_{an_a(N)}^h \end{bmatrix} \in R^{n_a \times N} \tag{7}$$

where $w_{a_{kl}}^h$ is the weight connecting kth input to the kth hidden neurons. The matrix of the weights between the hidden and output layer is defined as:

$$W_a^o = \begin{bmatrix} w_{a11}^o & \cdots & w_{a1(m)}^o \\ \vdots & \vdots & \vdots \\ w_{an_a1}^o & \cdots & w_{an_a(m)}^o \end{bmatrix} \in R^{n_a \times m} \tag{8}$$

where $w_{a_k}^o$ is the weight relating kth hidden layer to the output layer.

C. Learning Rules for Actor and Critic Networks

The gradient descent rules usually are used for updating the critic and actor network weights. The direction for the critic network is toward minimizing bellman error, and for the actor, the objective is to minimize the value function estimation. In the following, the update rules for the actor and the critic are illustrated separately.

1. Critic Training

If the Bellman prediction error is defined as:

$$e_c(t) = c(t) + \gamma\widehat{J}(t+1) - \widehat{J}(t) \tag{9}$$

where the reinforcement signal is defined by:

$$c(t) = e(t)^T Q_c e(t) \tag{10}$$

then the Bellman error will be defined as:

$$E_c(t) = \frac{1}{2}e_c^2(t) \tag{11}$$

The NN weights are then tuned by the gradient descent algorithm as:

$$\widehat{W}_{t+1} = \widehat{W}_t + \Delta W, \qquad \Delta W = -\boldsymbol{\alpha}\frac{\partial E}{\partial W_t} \tag{12}$$

where $\boldsymbol{\alpha}$ is the learning rates. For the critic network, which approximates the value function and is the function of the tracking error, the tuning laws apply as:

$$\hat{w}_{c_{ij}}^h(t+1) = \hat{w}_{c_{ij}}^h(t) - \boldsymbol{\alpha}_c\frac{\partial E_c(t)}{\partial \hat{w}_{c_{ij}}^h(t)} \tag{13}$$

$$\hat{w}^o_{c_i}(t+1) = \hat{w}^o_{c_i}(t) - \boldsymbol{\alpha}_c \frac{\partial E_c(t)}{\partial \hat{w}^o_{c_i}(t)} \tag{14}$$

Applying the chain rule for gradient descent approach for the hidden layer gives:

$$\frac{\partial E_c(t)}{\partial \hat{w}^h_{c\,ij}} = \frac{\partial E_c(t)}{\partial e_c(t)} \frac{\partial e_c(t)}{\partial \hat{J}(t)} \frac{\partial \hat{J}(t)}{\partial \boldsymbol{\phi}_{c_i}(t)} \frac{\partial \boldsymbol{\phi}_{c_i}(t)}{\partial \hat{w}^h_{c\,ij}} \tag{15}$$

same for output layer gives:

$$\frac{\partial E_c(t)}{\partial \hat{w}^o_{c_i}} = \frac{\partial E_c(t)}{\partial \hat{J}(t)} \frac{\partial \hat{J}(t)}{\partial \hat{w}^o_{c_i}(t)} \tag{16}$$

The convergence for this update rule can be tricky due to the nature of the approximation with this gradient descent method.

2. Actor Training

The objective of the actor network is the minimization of the approximated value function. Therefore, the difference between the evaluated function approximation $J(t)$ and the pre-defined value-function goal $J^*(t)$ (e.g. designed by expert knowledge considering the chosen cost function $c(x)$) is used for defining the actor error $e_a(t)$ as:

$$e_a(t) = J(t) - J^*(t) \tag{17}$$

The square actor error is defined as:

$$E_a(t) = \frac{1}{2} e_a^2(t) \tag{18}$$

The same gradient descent tuning laws can be applied for the actor network as:

$$\hat{w}^h_{a_{kl}}(t+1) = \hat{w}^h_{a_{kl}}(t) - \boldsymbol{\alpha}_a \frac{\partial E_a(t)}{\partial \hat{w}^h_{a_{kl}}(t)} \tag{19}$$

$$\hat{w}^o_{a_k}(t+1) = \hat{w}^o_{a_k}(t) - \boldsymbol{\alpha}_a \frac{\partial E_a(t)}{\partial \hat{w}^o_{a_k}(t)} \tag{20}$$

where $\boldsymbol{\alpha}_a$ is the actor learning rate. The chain rule for actor gradient descent algorithms is derived for hidden layer as:

$$\frac{\partial E_a(t)}{\partial \hat{w}^h_{a_{kl}}(t)} = \frac{\partial E_a(t)}{\partial e_a(t)} \frac{\partial e_a(t)}{\partial \hat{J}(t)} \frac{\partial \hat{J}(t)}{\partial x(t)} \frac{\partial x(t)}{\partial u(t)} \frac{\partial u(t)}{\partial \hat{w}^h_{a_{kl}}(t)} \tag{21}$$

and for the output layer weights:

$$\frac{\partial E_a(t)}{\partial \hat{w}^o_{a_k}(t)} = \frac{\partial E_a(t)}{\partial \hat{J}(t)} \frac{\partial \hat{J}(t)}{\partial \hat{w}^o_{a_k}(t)} \tag{22}$$

The simultaneous update of the actor and the critic make convergence guarantee more challenging to prove [22]. To ensure sufficient exploration within the states of the system and convergence to the global optimum, the control input signal should satisfy the persistence of excitation (PE) condition. Some of the PE methods can be a simple white noise, a filtered noise signal or a sinusoidal sweep added to the control input as an action modifier.

The approach proposed above can be seen as a generic actor-critic approach in solving optimal tracking control problem [15, 22, 23]. The convergence of actor-critic methods is provided in [24, 25].

3. Model Network

To derive the control value at the next time step, in HDP control approaches, a model of the system, is required. This model can be updated at each time step based on the target states and the current values for the states. Therefore, it adopts incrementally and is only valid locally. If it is assumed that a third neural network is used for the plant approximation, the input to the plant is the current state estimation, and the action derived by the actor and the output is the estimated states. The update rule for the model network is based on the error given by:

$$e_m(t) = x_{target}(t) - x(t) \tag{23}$$

The square of the model error is given by:

$$E_m(t) = \frac{1}{2} Q_m e_m^2(t) \tag{24}$$

where Q_m is the adjustment matrix for the model. The update rules for the model then will be the same as the critic network, and to avoid duplication will not be discussed here.

III. The Innovative Control Effector Aircraft Model

The continuous RL method of HDP was used to develop a flight control system for the ICE aircraft in a simulation framework. Lockheed Martin designs this highly maneuverable tailless aircraft with an unconventional control suite configuration, which results in substantial control challenges. In this section, details about the ICE configuration and model are given.

A. ICE Aircraft Control Objectives and Challenges

The ICE aircraft designed control objectives are mainly based upon maneuverability, stealth, and resilience. The ICE aircraft configuration is designed with Relaxed Static Stability (RSS) in both pitch and yaw axes to achieve high maneuverability. From 1974, the next generation of highly maneuverable fighters introduced that take advantage of the inherently aerodynamic instability to get extraordinary agility while relying on FCC to stabilize the aircraft for the human pilot. Flying an aircraft with RSS is achieved by fitting stability augmentation devices and with the constant intervention of the FCS to maintain the aircraft level. Therefore, the continuous intervention of an automated FSC is also critical and vital for ICE design. ICE aircraft is designed without horizontal and vertical tails. In a tail-less aircraft, other advantages can be obtained apart from low RCS, as the flying-wing configuration is very efficient in producing lift, allowing for overall savings in weight and cost. Without the tail, the stabilizing moments in pitch and yaw for ICE have to be generated with the proper actuation of the redundant control surfaces, a challenge defined in the context of the ICE development objectives [3]. The ICE aircraft is also meant to be categorized as Unmanned Air Vehicles (UAVs). Therefore, improvements in automatic adaptive control systems are essential for this innovative design. In the occurrence of accidents such as the signal loss and other failures, adaptability and the ability to control the aircraft remain completely to the smart FCC on-board.

B. ICE Aircraft Control Suite

The land-based baseline configuration of the ICE simulation model is described by [26] as a supersonic single-engine, 65 deg sweep delta flying wing and a multi-role fighter. The ICE aircraft has highly redundant 13 multi-axis control effectors [26]. A schematic of the control suits for ICE is presented by Fig.4. The control effectors include Leading-edge flaps (LEF), all moving wing tips (AMT) and spoiler-slot deflectors (SSD) which are mainly used for lateral-directional control, the pitch-flaps (PF) to provide longitudinal control power and elevons and multi-axis thrust vectoring (MTV) that are used for both symmetric and asymmetric movements. The elevons that can deflect independently and asymmetrically for roll control and also contribute to pitch control by symmetric deflections. Two pairs of LEfs use for lateral-directional control in high angle of attacks (AOAs) and low speeds. AMTs are the primary sources for yaw control in high angle of attacks and can maintain their effectiveness for wide ranges in the speed envelope of the aircraft. The SSDs are more useful for yaw augmentations in a lower angle of attacks.

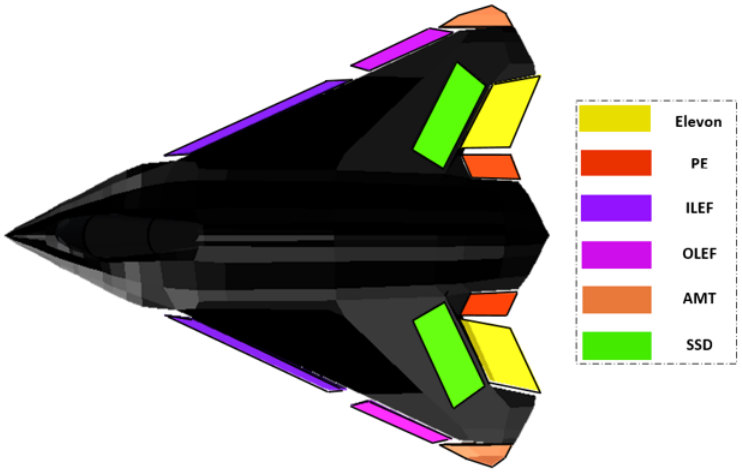


Fig. 4 ICE aircraft control suite

Integrating thrust vectoring power to the control power substantially improves the potential of the aircraft for unlimited maneuverability. MTV is designed for both pitch, and yaw thrust vectoring for ICE aircraft and also allows the full utilization of the roll control power by integration with SSDs and by coordinating the yaw control. In Table 1, the list of the control suites as well as their position and rate limits are described.

Table 1 ICE aircraft control suite specification

Effector	Position limits [deg]	Rate limits [deg]	Positive direction
Elevon	[-30, 30]	150	Downwards
PF	[-30, 30]	150	Downwards
ILEF	[0, 40]	40	Downwards
OLEF	[-40, 40]	40	Downwards
AMT	[0,60]	150	Downwards
SSD	[0, 60]	150	Upwards
MTV	[-15,15]	150	[-]

C. Three Degree of Freedom Aerodynamic Model

A spline-based model developed by [27] for ICE aircraft is used for calculating the force and moment coefficients in the MATLAB environment. In the spline model the aerodynamic coefficients are identified with a spline approximation as a function of the $(\alpha, M, \beta, V, p, q, r, T, u)$, where $V(ft/s)$ is the true airspeed, p, q and r (rad/s) are the body rotation rates in x, y and z axes, $T(lbs)$ is the thrust and u is the input vector given by:

$$\begin{aligned}\mathbf{u} =&[\delta_{ILEF_L}, \delta_{OLEF_L}, \delta_{AMT_L}, \delta_{E_L}, \delta_{SSD_L}, \delta_{PF},\\ &\delta_{ILEF_R}, \delta_{OLEF_R}, \delta_{AMT_R}, \delta_{E_R}, \delta_{SSD_R}, \delta_{pTV}, \delta_{yTV}]\end{aligned} \tag{25}$$

where L and R denotes to the left and right effector and pTV and yTV corresponds to the thrust vectoring for pitch and yaw movements. As this study only considers longitudinal motions of the aircraft, only the equations correspond to the symmetrical moves are described here. Also, to simplify, the control surfaces deflections are assumed to be symmetric and equal. The yaw thrust vectoring value, too, is set to zero for the whole simulation. Including the thrust vector projection on the body frame F^b in the derivation of the aerodynamic forces and moments, the aerodynamic coefficients are obtained by the model and the longitudinal forces and moments are computed by:

$$F_x = Tcos(\delta_{pTV})cos(\delta_y TV) - \frac{1}{2}\rho V^2 S C_{F_x} \tag{26}$$

$$F_z = -Tsin(\delta_{pTV})cos(\delta_y TV) - \frac{1}{2}\rho V^2 S C_{F_z} \tag{27}$$

$$M_y = \frac{1}{2}\rho V^2 S\bar{c} C_{M_y} - Td_{TV} sin(\delta_{pTV})cos(\delta_{yTV}) \tag{28}$$

where d_{TV} is the thrust vectoring arm from aircraft center of gravity and the forces and moments are in reference body frame F^b in $[lbf]$ and $[lbf.ft]$, respectively. Euler equation is used for computing the changes in the system dynamics. The atmospheric changes and conditions are described based on the International Standard Atmosphere (ISA).

IV. The Adaptive Critic Design-Based Flight Control System

The ICE aircraft spline model is used as the environment. The ICE environment is implemented using MATLAB script to simulate the dynamics of the designed aircraft. The thrust is controlled by an internal Proportional and Integral (PI) speed controller in the simulation model. The lateral-directional movements of the aircraft are not considered, and only longitudinal changes in the dynamics are considered for the angle of attack track. Also, it is assumed that the perturbations in the longitudinal and lateral direction are independent, and their variables are not coupled, which is valid for an aircraft of symmetric shape under a steady equilibrium condition. The state vector for the longitudinal control of the ICE aircraft is defined as Eq. (29), including the position, the pitch rate, airspeed, aerodynamic and Euler angles.

$$x^{ICE} = [\dot{x} \;\; \dot{z} \;\; x \;\; z \;\; u \;\; w \;\; \alpha \;\; \theta \;\; q \;\; V \;\; M \;\; \rho \;\; \bar{q}]^T \tag{29}$$

A longitudinal nonlinear attitude control system based on the available spline aerodynamic model with a heuristic dynamic programming RL controller was designed for the ICE aircraft. Such an architecture for the control system is known as Adaptive Critic Designs (ACDs). The HDP framework discussed earlier is used to perform a reference tracking control problem for the ICE aircraft. The aim is to explore the tracking performance of the proposed HDP framework for the over-actuated system. First, the angle of attack tracking problem is investigated with multiple benchmarks starting with one effector (e.g. pitch flaps) as the control inputs available and adding more effectors till all the effectors (except the thrust vectoring) become available for the system. The performance of the controller in the tracking task for the inner loop attitude control and its exploitation of the available control inputs with its effect on the tracking performance will be explored both qualitatively and quantitatively. As only longitudinal control is considered, the control effectors' deflections are assumed to be symmetric to avoid the lateral-directional movements.

A. Angle of Attack Tracking Problem

The ACD controller is implemented for the inner control loop of longitudinal body-frame angle control, so the angle of attack of the aircraft. The RL controller uses the pitch flap control surfaces first to provide angular rate control in the longitudinal axis. There is no direct control for the angular rates. For the model identification, however, both the angular positions and velocity are fed to the network. In this way, the RL controller learns the direct effect of the actuators on the angular position and rates and the aircraft dynamics can be learned quickly. This RL controller can then be used to provide a higher level of control objective (e.g. altitude control) using other control techniques (e.g. PID controllers or nonlinear dynamic inversion techniques), as will be shown by V.

Fig.5 gives a general overview of the inner control framework, including the higher-level overview of the continuous RL controller. As shown, the architecture consists of the designed RL controller and the ICE plant. In the implementation of the HDP method for the ICE aircraft, the same framework, as described by section II, is used. The controller includes three main entities, namely the critic, actor and the model of the system. In the above framework, the actor learns the control policy and develop the control law while the critic updates the value function and gives the direction to the actor towards the optimal policy. To perform back-propagation of the actor network, online system identification is used to provide the derivatives and state predictions. Three neural networks were used for the critic, actor and model networks with gradient descent method used for propagation of the information as shown by dotted lines in Fig.5.

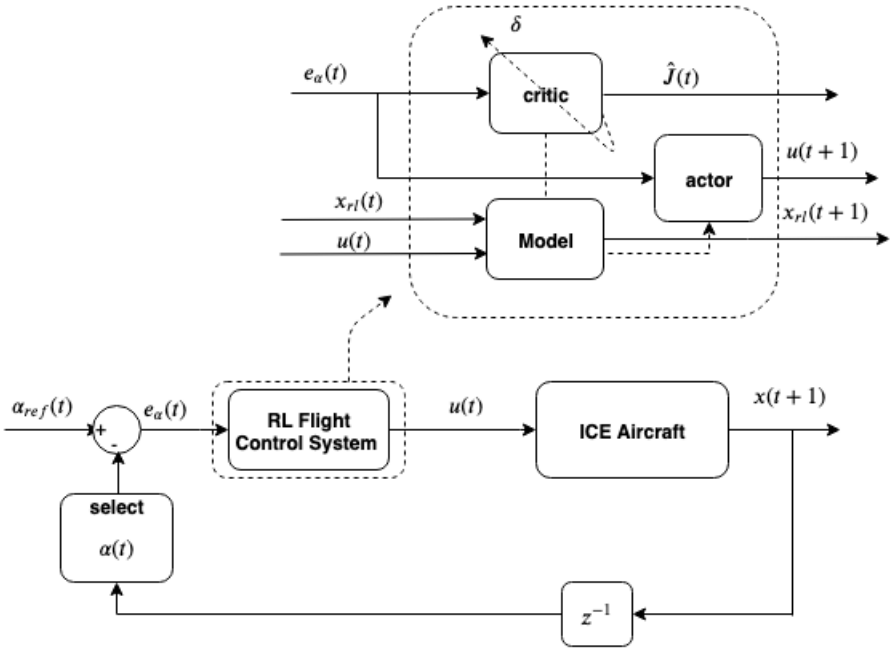


Fig. 5 Layout of the control loop including the reinforcement learning controller for attitude control of the ICE aircraft

The angle of attack response is selectively used to find the error from the response and the reference signal to generate the inputs to the critic and the actor. Therefore, the controller receives the angle of attack as the control state for each time step, $x(t)$, in addition to the reference signal, $x_{ref}(t)$. The reference vector includes only one component of the angle of attack, α_{ref}. The value of the angle of attack reference is provided by the designer and defined by Eq. (30) as a sinusoidal signal initializing at the angle of attack of 8.6 degrees.

$$\alpha_{ref}(t) = \alpha_{trim} + 5sin(t) \quad (deg) \tag{30}$$

To provide the RL controller with the information on the tracking task and since an accurate model of the system is not required for the HDP architecture, the reinforcement learning states $x_{rl}(t)$ are selected from the system states $x(t)$ as defined by Eq. (31). This information from the states will be used for updating the value function, the policy and, ultimately, the control inputs.

$$x_{rl}(t) = \begin{bmatrix} \alpha(t) \\ q(t) \end{bmatrix} \tag{31}$$

In order to get a faster learning performance, from the RL states, only the angle of attack is used to obtain the tracking error that will be used as the inputs for the actor and critic networks. In this way, the actor and the critic receive information on the current systems as well as the goal. The tracking error is given by Eq. (32).

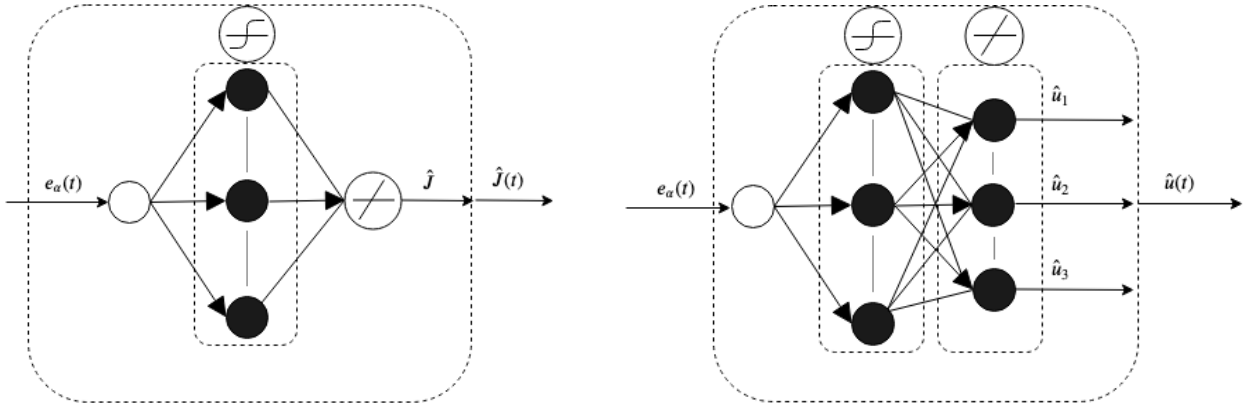
$$e_\alpha(t) = \alpha(t) - \alpha_{ref}(t); \tag{32}$$

To update the controller in the direction that the system moves towards the current system reference and maintain that, the controller requires the prediction of the successive reinforcement learning state $\hat{x}_{rl}$ that comes from the model and is used for the update of the value function.

B. The Neural Networks Design

In Fig. 6, a visualization of the actor and critic designs are shown. Both the critic and the actor networks consist of a single fully connected neural network. The critics map the state error to the value corresponds to that system state while

the actor assigns the state error to the control action. The number of outputs of the actor will be initially set based on the number of available control effectors.



(a) **(b)**

Fig. 6 Schematic of the proposed (a) critic and (b) actor networks

For the model of the system, again, a fully connected neural network is used with its structure shown by Fig. 7. The model network inputs increase based on the available control inputs to the system. Therefore, an augmented vector will be used as the input to the model network.

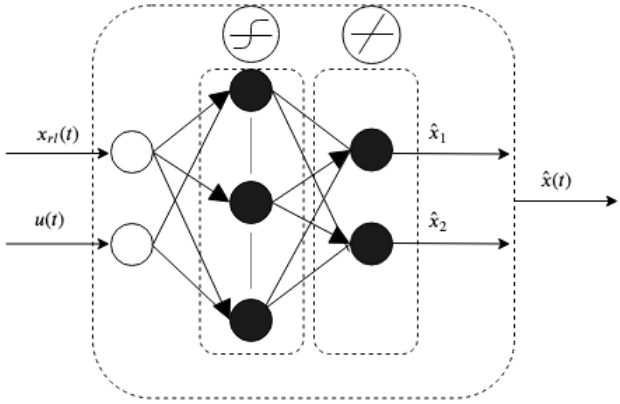


Fig. 7 Schematic of the proposed model network

C. The Learning Rates Scheduling

In an HDP framework, the actor and the critic are trained iteratively and simultaneously, which results in the online learning ability. In other words, the weights are updated incrementally at each time step based on the current error. The learning rates are very important in the convergence to the optimal network weights. Therefore, the proper choice for these parameters is crucial for convergence time and to avoid local optimum trapping. For the critic, actor and model networks, adaptive learning rates are used [14], with the update rule defined by Eq. (33). To avoid unwanted parameter variations in online learning, the learning rates scheduling is only performed in the first few seconds and are set constant afterwards.

$$\alpha(t+1) = c_\alpha \alpha(t) \tag{33}$$

where c_α is the learning rate coefficient.

D. The Networks Initialization

For any RL approach, including the HDP method, hyper-parameters are set for the controller. These parameters include learning rates, α and discounting factor γ. The number of neurons and the weight initialization are also crucial in the convergence of the controller. The learning parameters for the RL controller are shown by the table below. The weights of the networks are initialized using symmetric randomization. The initial learning rates are mentioned in the table below. During the online learning, these parameters, except for the learning rates, are kept constant to maintain the converged policy in a stationary condition.

Table 2 The RL controller learning parameters

Parameter	Value	Parameter	Value	Parameter	Value
$\boldsymbol{\alpha}_{a_0}$	500	γ_c	0.995	c_{α_a}	0.9
$\boldsymbol{\alpha}_{c_0}$	1	neurons	25	c_{α_c}	0.8
$\boldsymbol{\alpha}_{m_0}$	0.15	W_0	[-1, 1]	c_{α_m}	0.99

The model matrix which is adjusted by trial and error is defined as:

$$Q_m = \begin{bmatrix} 10.5050 & 0 \\ 0 & 3.2423 \end{bmatrix} \tag{34}$$

This weight matrix is used to adjust the focus of the model learning on the angle of attack rather than the pitch rate. Although both important, however, the RL controller task is set for tracking the angle of attack and more information for the controller is required for this state. Therefore, the accuracy of the model identification for this state becomes more crucial that can be implemented by giving more weights to the angle of attack updates.

E. Cost Function

The reward or cost definition highly depends on this environment. The configuration of the ICE model was described in section III. Since the goal is to track the reference signal for the angle of attack α_{ref}, a cost function is designed in a way to return the optimal value of zero for tracking the reference signal precisely, in any other case, the cost value will be a positive value. Therefore, the cost function is designed for this case as given by Eq. (35).

$$c(t) = Q_c e_\alpha^2(t) \tag{35}$$

where Q_c is the cost weight matrix and is defined as:

$$Q_c = \frac{200}{\alpha_{max}^2} \tag{36}$$

F. Research Cases for Attitude Control

To compare the performance of the controller for different effector configurations, a benchmarking set up is considered. The same initial conditions are used for the controller in each situation. The aerodynamic control surfaces considered for the four bench markings are shown by Table 3. These control surfaces are used as controllable inputs to the model, which includes the commanded pitch flaps in the beginning and add the elevons, the inboard and outboard leading edge flaps, and for the last benchmark, all the effectors eliminating the yaw and pitch thrust vectoring are considered. It is assumed that the effectors deflect symmetrically to eliminate their effect in lateral-directional movements. The performance of the controllers is compared in terms of their tracking performance, their computational effort, the controller stability and the effectors' control effort.

Table 3 Attitude control benchmarking cases

Case	Control Inputs
BC1	δ_{PF}
BC2	$\delta_{PF}, \delta_{Elevon}$
BC3	$\delta_{PF}, \delta_{Elevon}, \delta_{ILEF}, \delta_{OLEF}$
BC4	$\delta_{PF}, \delta_{Elevon}, \delta_{ILEF}, \delta_{OLEF}, \delta_{AMT}, \delta_{SSD}$

V. Outer Loop Altitude Control

To show the advantage of the RL controller in performing more difficult control objectives such as altitude tracking, an altitude control outer loop is implemented. The PID controller receives z_{ref} and selected $z(t)$ as inputs. The gains for the PID feedback controller are set manually and using trial and error. In contrast to the previous definition for the controller, the values of the angle of attack reference are provided from the PID controller. An overview of the structure of the complete control framework is shown by Fig.8. The framework of the RL controller is the same as was shown in the previous section.

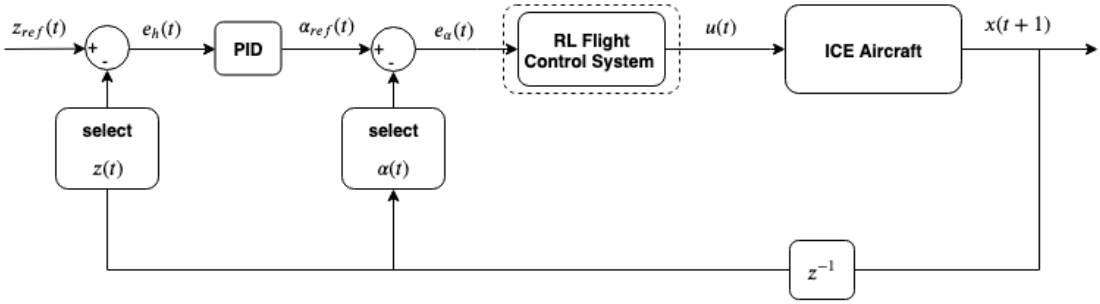


Fig. 8 Layout of the altitude control framework including the RL and the outer loop PID controllers

The controller starts with an offset from the desired altitude, and the goal is to reach the goal altitude and stabilize there (a regulation problem). The z_{ref} is defined by the designer as a constant value of:

$$z_{ref} = -10000(ft) \tag{37}$$

A. Research Cases for Altitude Control

The PID control values are presented for three different altitudes benchmarking conditions of BCH11, BCH12 and BCH22. In the first two cases, the two effectors of pitch flap and elevons are available for the controller, and only the proportional gain is changed. For the last case, all the effectors are used except for the thrust vectoring. Table 4 shows these conditions and the PID gains associated with each situation. The saturation limits are set according to the minimum, and maximum angle of attack values derived from the ICE aircraft flight envelop.

Table 4 Altitude control benchmarking cases

Case	Control Inputs	$K_P(10^{-4})$	$K_I(10^{-5})$	$K_D(10^{-3})$
BCH11	$\delta_{PF}, \delta_{Elevon}$	5.8	2	1
BCH12	$\delta_{PF}, \delta_{Elevon}$	2.6	2	1
BCH22	$\delta_{PF}, \delta_{Elevon}, \delta_{ILEF}, \delta_{OLEF}, \delta_{AMT}, \delta_{SSD}$	2.0	2	1.5

The framework for the control loop discussed in this section is used for online altitude control of the ICE aircraft.

VI. Simulations and Results

In this section, the simulation setup and online results for the ICE aircraft attitude control using the ACD controller and the altitude control with integration of the ACD and PID controller are given separately. The controller and the dynamics of the aircraft are simulated in the MATLAB environment. The dynamics of the aircraft is updated using the longitudinal force and moment coefficients that are derived from a spline model [27]. The forces and moments are then derived from these coefficients (subsection III.C). The states are updated using the Eulers method.

A. ICE Aircraft Environment

The simulation is operating with a frequency of $1000Hz$ and does not consider the turbulence effects. The measurement errors and sensor dynamics are not included in the output states. To add exploration to the system, persistent excitation in the form of white noise is added to the control inputs. The initialization of the ICE aircraft is performed in the flight condition of -1000 ft altitude and $432 ft/s$ airspeed for the attitude control, with the initial values for the thirteen effectors set for their trim values for the specified altitude and airspeed. The heavyweight condition of the ICE aircraft is considered for the simulation. The ICE properties are shown by Table 5.

Table 5 The ICE aircraft properties

Property	Value	Unit
mass	1152.6	$[slug]$
S_{ref}	808.6	$[ft^2]$
$\bar{c}$	28.75	$[ft]$
I_{yy}	81903	$[slug.ft^2]$
g	32.174	$[ft/s^2]$

A Proportional-Integral (PI) thrust controller is used for airspeed control [10]. The reference airspeed is defined as the initial airspeed value, so 430 ft/s. The thrust controller proportional gain is set equal to the mass of the aircraft, while the integral gain is set to 10. The thrust values are saturated between 0 and 10, 000 lbf with anti-windup added for saturation of the integral control signal to the maximum thrust.

B. Initialization

For the attitude control part, the aircraft is initialized at the altitude and airspeed mentioned above, while the angle of attack, the pitch angle, and the pitch rate are set to values different from the trim. The aircraft is highly unstable, mainly if the trim inputs are not applied. Therefore, the control problem becomes very challenging for the HDP controller. Without a suitable controller, the states can easily deviate from initial values. The same state initialization is used for all the presented experiments in this paper for the sake of comparison except for the altitude control problem where the aircraft starts with an offset from the desired altitude with other states' initialization remain the same.

C. Attitude Control Results

The online attitude control lasts for 60 seconds. In this part, the online results are shown. The results displayed here are corresponding to one of the successful runs of the online controller. At each simulation step, the ICE model receives the reinforcement learning state vector x_{rl} and the control inputs vector $u(t)$ and predicts the response of the reinforcement learning states (state estimation). Since the initial visits in the state and action space are essential for the controller faster convergence, the persistent excitation is only applied for the first 10 seconds. Fig.9 shows the progress of the controller

as more experience is obtained during online learning. The figure shows one of the best recorded performance with the dashed line showing the angle of attack desired trajectory. The top part of the picture shows the error values for the critic (top left) and actor (top right). The plots illustrate the progress of the networks with the initial peaks corresponding to the initial exploration. The figure presents high critic error in the first 5 seconds. The same tracking error can be seen for the tracking performance by just looking at the angle of attack response, and the reference signal in the middle left part of the figure. The actor errors are small in the beginning and become very large between 1 to 3 seconds. However, the controller can adapt very fast in the very first few seconds. Fig. 10 shows the attitude time history of the controller, with the first seconds zoomed in. The figure confirms that the controller has learned to track the reference trajectory precisely in less than 5 seconds.

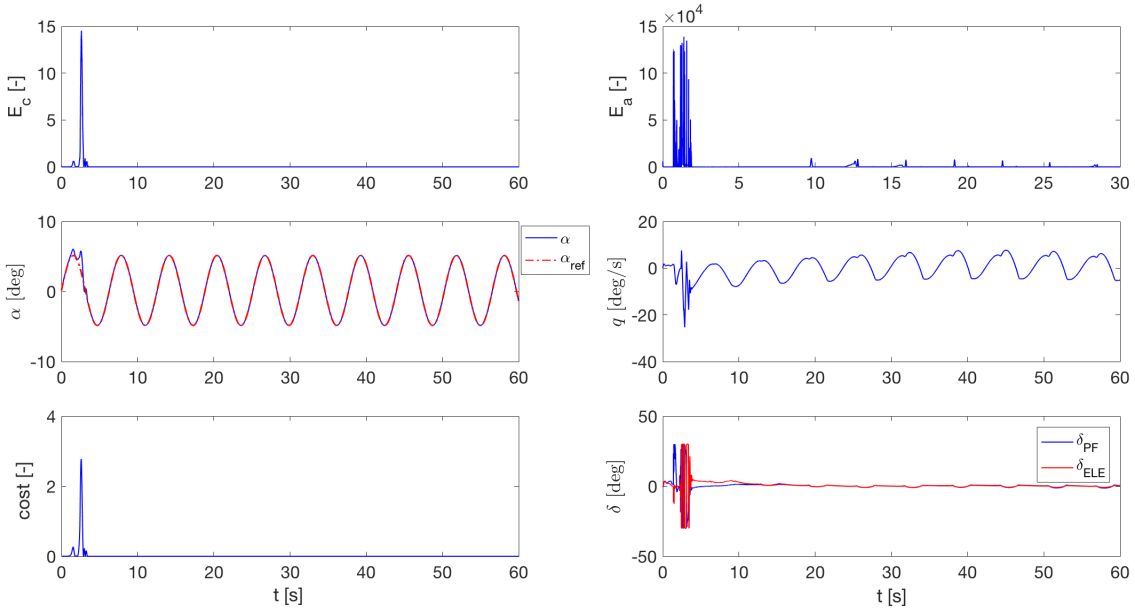


Fig. 9 The attitude control results with the critic network error in top left, the actor network error in the top right, the angle of attack response and the reference angle of attack in the middle left, the pitch rate response in the middle right, the cost value in bottom left, the effectors deflection in the bottom right (case BC2)

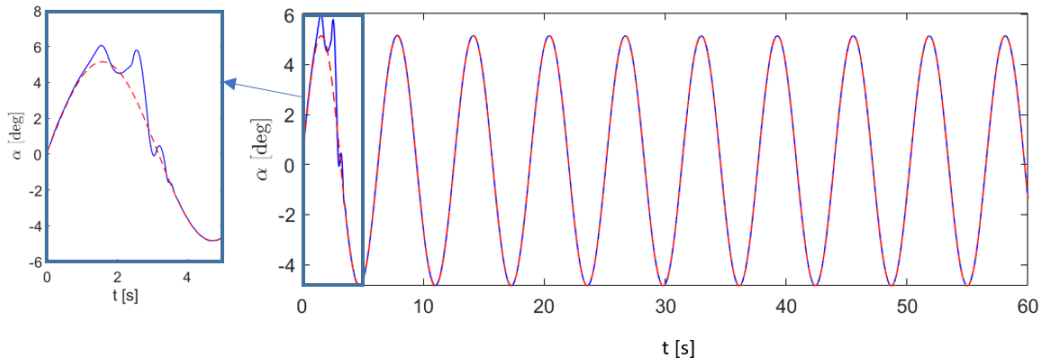


Fig. 10 The time history of the angle of attack response and reference signal with the first 5 seconds zoomed in (case BC2)

The pitch rate response is shown in the middle right part of the graph. The response shows variations with less frequency in the first few seconds, and these changes indicate a higher frequency after 2 seconds, while after 5 seconds, the pitch rate response becomes oscillatory with some small changes in the periodic behavior over time. The left bottom

plot represents the cost values, which shows that the agent successfully progresses toward minimization of the cost, except for the first 3 seconds of exploration. As the cost is only a function of the states, the cost values show peaks with the same frequency as the tracking error. The cost values indicate that the controller can adapt to the reference trajectory within a few seconds. The bottom right shows the pitch flap and elevon deflections. The control inputs are saturated according to the limitations mentioned in the configuration. It can be seen that the controller first applies smaller contributions, starting from their initial values. The effectors are pushed to their limits with the increase in the actor error; however, after some exploration, the controller obtains less aggressive control signals. It shows that the actor networks are continually changing for a better policy over the 60 seconds. The significant error for the actor shows its deterioration effect on the pitch flap and elevons deflections that vary with high frequency in the same duration.

The same high frequency but high amplitude variations can be seen in the control input response, which reduces the performance. Since the control effector deflections are not considered in the cost function, there is no direct information obtained by the RL controller for the control effector deflections' influence on the cost function. However, the controller can converge quickly to a smoother response when the high tracking performance is achieved. While the controller can perform the tracking task accurately from the first few seconds, the policy still shows a few changes in the learning period.

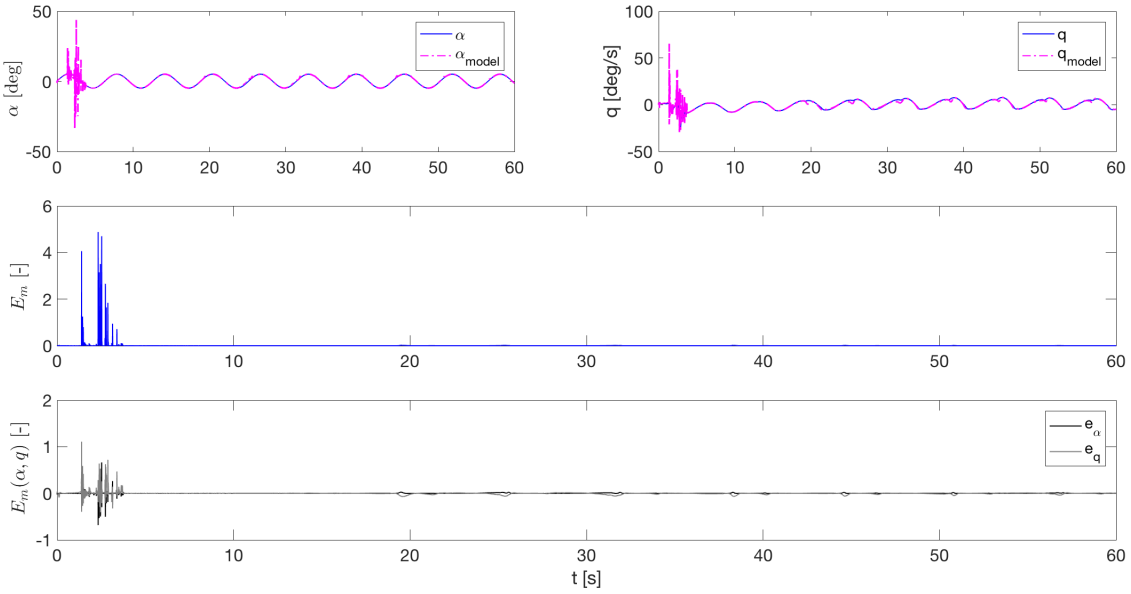


Fig. 11 The online model identification results with the angle of attack and pitch rate plant and model responses on top, the overall model error in middle and the angle of attack and pitch rate errors in the bottom (case BC2)

Fig. 11 shows the online model identification results. The top left and right top parts of the plot shows the identification of the angle of attack and pitch rates, respectively. The dotted lines show the model identified states while the solid lines show the response of the ICE plant. The model is identified precisely for the angle of attack after 5 seconds, while for the pitch rate, there is some deterioration at some points in the identification process. The same observation can be made looking at the model error for the pitch rate in the bottom figure. The reason can be the low learning rate value for the model network and the weight matrix for model identification. However, the overall model error shows a great improvement after 5 seconds. The zoomed-in figure for the identified model in Fig. 12 shows that the identification improves over time. The identification errors agree with continuous change in effectors deflections for the whole learning period as part of the actor training depends on the derivative of the identified model states to the effector inputs.

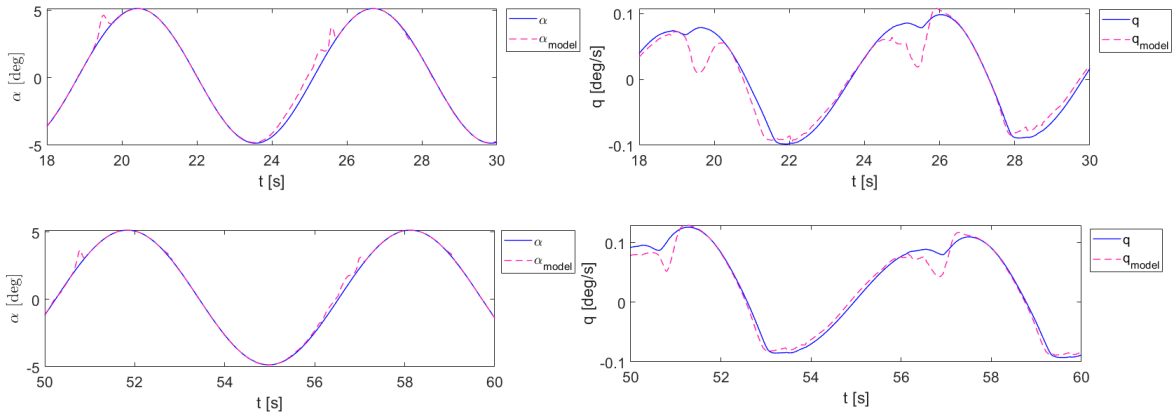


Fig. 12 The zoomed-in (case BC2) plot for model identification for the angle of attack and pitch rate between 18 to 30 seconds (top plots) and 50 to 60 seconds (bottom plots)

Fig.13 indicates the changes in the network weights for the controller. The convergence of the critic networks agrees with what already was shown by the tracking performance of the controller. However, the actor hidden layer networks show some changes within time, while the output layer weights show convergence. Therefore, the controller is still updating the policy during online learning, as was also observed from the figures above. The actor weights show more significant changes between the first 20 seconds, which was also seen in the policy obtained and presented with pitch flap and elevon deflections. The model weights show variations through online learning, which is expected as the model identification is performed locally. However, it is interesting to note that the PE added to the system initially is identified quickly by the controller and not influenced the model weights.

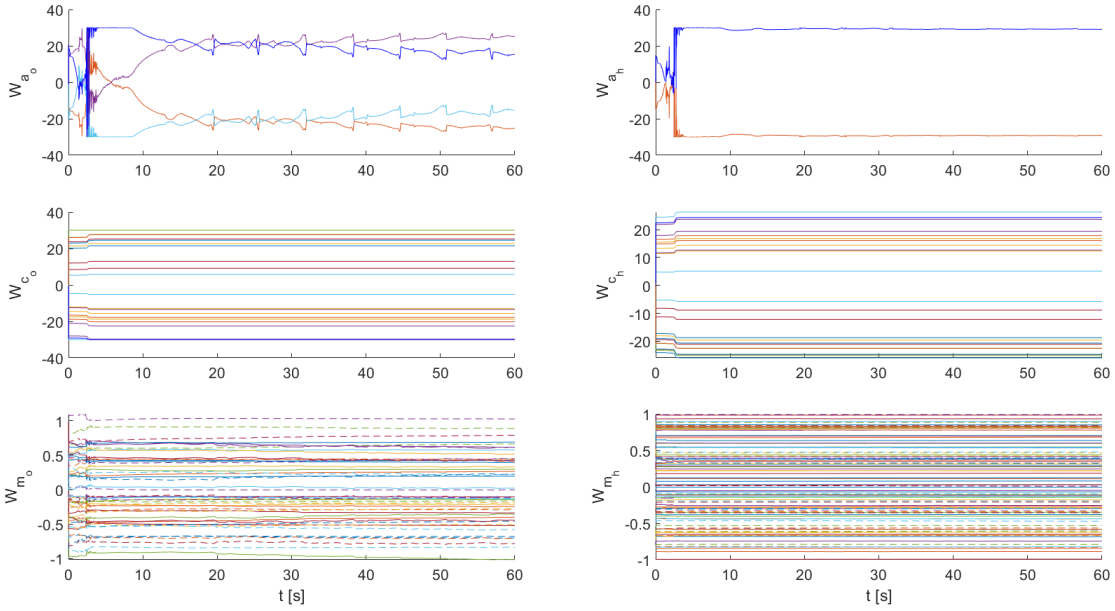


Fig. 13 The weights updates for the critic, actor and model networks (case BC2)

Qualitatively speaking, by looking at the graphs presented above, it was observed that the controller can adapt to the reference signal, and the controller is able to follow the reference angle of attack precisely in the first few seconds with online training. To see the control allocation performance of the HDP controller, in this case, the BC2 control

effectors response is presented by Fig.14, pitch flaps and elevons are the control effectors that have the most influence in longitudinal pitch control of the ICE aircraft (without thrust vectoring). Therefore, it is interesting to see how the control effort is distributed between these effectors.

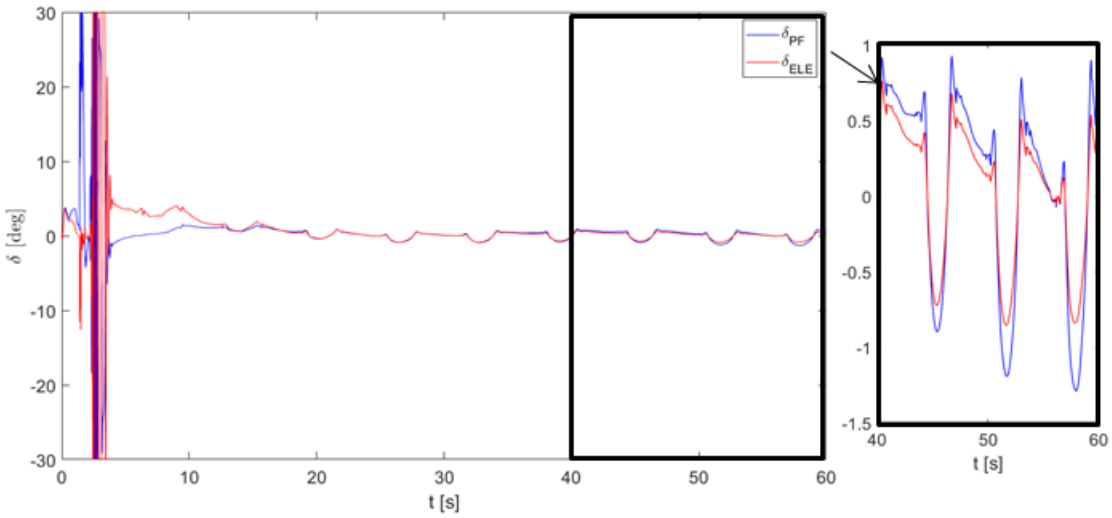


Fig. 14 The online progress of the control policy for 60 seconds of learning and the zoomed in response for the last 10 seconds (case BC2)

Fig.14 shows that the controller is capable of achieving a policy that uses the power for both effectors to achieve the angle of attack control task and the effectors' deflection are contributing in the same direction. The current performance is made while the cost function definition does not provide any feedback for the control effort, and the controller performs the angle of attack tracking objective based just on the tracking error information.

D. Comparison of the Tracking Performance

In the previous section, one of the benchmark conditions with the best performance was presented. The quantitative performance presentation for all the research cases is shown in this section to provide a basis for the comparison. The values for comparing criteria correspond to one of the successful runs. To compare the tracking performance for each condition, the Root Mean Square (RMS) of the tracking error is shown by Table 6. In addition, the simulation run time is shown to compare the computational cost of each condition. For the experiment runs, a PC with i5-6500 CPU and 8 GB of RAM is used. The control effort for each control surface is shown by Table 7, while Mean Control Effort (MCE) is defined by Eq. (38).

$$MCE = \frac{\sum |\delta_{eff}/\delta_{eff_{max}}|}{N} \tag{38}$$

where N is the number of samples during the online simulation. The success ratio is also provided for 100 runs for each condition with randomization in the initialization of the network weights to check for the stability of the controller.

Table 6 The learning performance for each benchmark condition

Case	RMS (α) [deg]	Run Time [seconds]	Success Ratio [%]
BC1	1.2843	28.9773	28
BC2	0.2251	29.7998	38
BC3	1.1277	42.4319	15
BC4	1.1589	124.5123	9

The above table results for the tracking error shows that the best tracking performance is obtained for the BC2 condition. Comparing the last two benchmarks, better tracking performance is achieved using all effectors in used when compared to the BC1. The tracking performance between the two last benchmarks is close to each other. The computational effort results are as expected, as the controller takes more time to learn the optimal policy when a higher number of effectors are used. The computational effort becomes much higher when all effectors are used, and both the model and actor networks become more complicated. The success ratios are low, which shows that the controller is quite unstable, and the success ratio becomes very low for the last benchmark. The instability of the controller can improve with some increase in the computational cost by adding more neurons to the actor and model networks. It was also found that the initialization of the network weights plays a vital role in the stability of the controller. As an example, a success ratio of more than 50 percent can be achieved if the controller weights are set to bigger initial values.

Table 7 The control effort for each benchmark condition

Case	PF MCE [-]	ELE MCE [-]	ILEF MCE [-]	OLEF MCE [-]	AMT MCE [-]	SSD MCE [-]
BC1	0.1690	0	0	0	0	0
BC2	0.0387	0.0462	0	0	0	0
BC3	0.0322	0.0407	0.0266	0.0203	0	0
BC4	0.0404	0.0468	0.0256	0.0185	0.0253	0.0122

Looking at the control efforts for pitch flaps, it can be seen that effort is reduced clearly when more effectors became available for the controller. This shows that the controller is successful in distributing the required control power between the available control effectors to follow the desired angle of attack trajectory. However, the control effort changes for pitch flaps become smaller for the cases of BC2 and BC3, while there is also a slight increase when all effectors are used when compared to the case where only four effectors are available to the controller. For elevons, the same trend can be seen as the control effort becomes smaller between the second and third benchmark condition but become higher for the last one. This can be due to the interaction between the effectors, especially for higher angles of attack that can cause more engagement of the control effectors. Also, by looking at the graphs, it was found that at the beginning of the learning, the controllers are pushed to their limits with higher frequency, while through the learning, they almost converge to a periodic response.

Another observation is the ILEF and OLEF control efforts for the last two benchmarks. The control effort for ILEF almost remains the same, while OLEF experiences lower control effort for the last benchmark compared to BC3. This can be due to the reason that the controller converges to another (near) optimal policy, and there is a different division of control power required between the control effectors. This can also include compensation of one effector for another. The control effort for the AMT and SSD effectors is quite small, which can be expected as these effectors have the most control power in lateral-directional movements.

E. Altitude Control Results

For the altitude control, the ICE aircraft starts at the same flight condition as for the attitude control, so airspeed of 430 ft/s with the same initial values for the effectors. However, in this case, the aircraft starts with -300 ft of offset from the desired altitude of -1000 ft. Therefore, the control problem is deemed more difficult if compared to a case where the

aircraft starts in the trim condition. The simulation runs at 1000 HZ with a maximum duration of 120 seconds. Different configurations for different available effectors, in addition to different PID controller gains, are tested as described by subsection V.A. The purpose is to see the ability of the outer loop controller to achieve the altitude tracking task while the RL controller is learning in parallel in an online framework. Fig.15 shows the altitude time history for the controller for three different research cases. The figure compares the approach behaviour for the regulation problem for three cases and shows how the controller behaves under different conditions. The results are shown for one of the best recorded performance. The dashed horizontal line in the figure shows the desired altitude. The two first cases only consider pitch flaps and elevons as the available control effectors, while the last case considers all the effectors except for the thrust vectoring. The altitude behaviour shows the success of the controller in moving toward the desired trajectory.

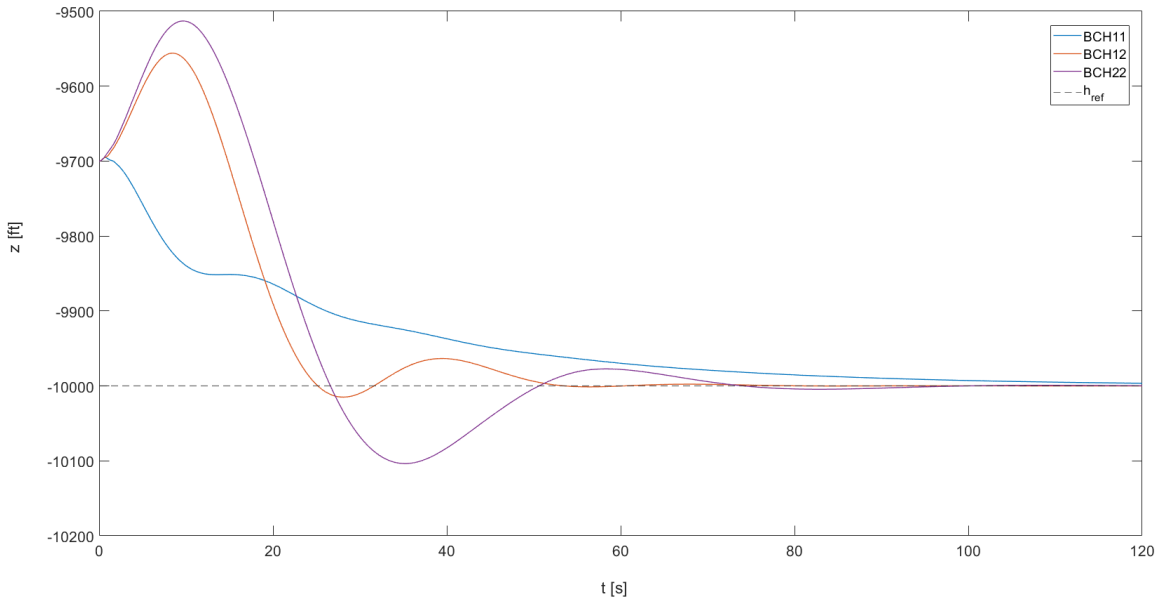


Fig. 15 The altitude tracking results for different benchmarking conditions with 300 ft offset

The altitude time history shows that the controller is capable of approaching the desired altitude and stabilizing the aircraft, especially for cases BCH12 and BCH22. Some differences can be seen for various conditions, as the chosen trajectory is expected to depend highly on the PID controller gains. The BCH11 case reflects a less aggressive manoeuvre and a smoother approach to the desired altitude. In other words, the variations in altitude are more gradual and with less frequency if any. While conditions BCH12 and BCH22 attempt to reach the reference altitude, there is an overshoot in the first few seconds, which become more prominent for the last case compared to BCH11. The 'rise time' is shorter for case BCH12 compared to BCH22, while there is an oscillation afterwards with a smaller amplitude around the desired trajectory. The overshoot can be explained due to the large initial angle of attack of the aircraft, which cause the behaviour of the aircraft first to start losing altitude and then recover from that and reaches the desired altitude and attempt to maintain that. However, it can be seen that by choosing a larger proportional gain, this overshoot can be very much damped while still happening. Comparing the first two cases, the more significant overshoot and more aggressive manoeuvre for the BCH12 compared to BCH11 result in stabilization of the aircraft around the desired altitude before 60 seconds. However, for the BCH11, the desired altitude is only reached after 150 seconds and slower. For the last case BCH22, the altitude stabilizes only after about 90 seconds, while the amplitude of the oscillations is larger. To get a better understanding of aircraft behaviour and to see if the aircraft is able to follow the obtained trajectory, the angle of attack of attack tracking and the effectors' policy should be analyzed.

Fig.16 shows the angle of attack response. The dashed lines in the figure show the reference angle of attack signals derived from the PID altitude controller. What the plot shows is that the angle of attack follows the reference signal precisely before 10 first seconds. While the zoomed-in figure also shows that the angle of attack is damped around the reference PID output after some initial oscillations. As could have expected for the BCH22 condition, the angle of attack tracking becomes accurate in a longer period as a more difficult task is defined for the controller. One interesting observation is that, as expected, the angle of attacks converge to a value close to the trim value at this altitude, where the

initial effectors were set based on. There are minimal deviations from the trim value, which is caused by the integration control signal. It was found from Fig.15 that the most significant result is obtained by case BCH11.

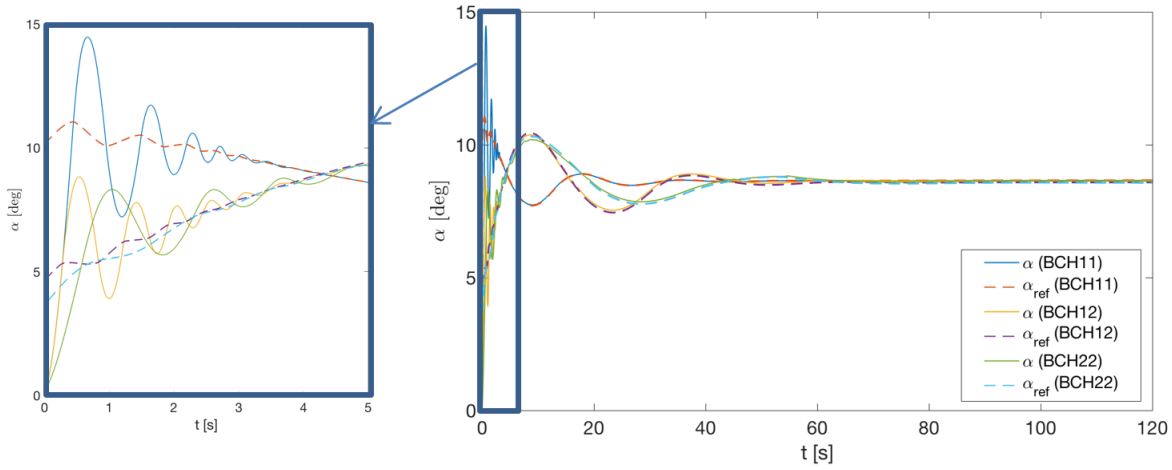


Fig. 16 The angle of attack reference signal and the response of the ICE plant for different benchmarking conditions with the outerloop altitude control

From Fig.16, it can be seen that for the BCH11 and BCH12, the controller performs more aggressive angle of attack tracking maneuver to increase the angle of attack at the start of the learning and adding to the thrust values to compensate for the decrease and drop in the airspeed as shown by Fig.17 and Fig.18. On the contrary, from the same figure, it can be seen that for the last case, the speed is initially increased very slightly, and therefore, the thrust decreased significantly while the angle of attack is expanding to reach the reference signal. After the angle of attack is damped around the reference signal, the velocity follows the same trend as the angle of attack behavior. As expected, since the integral gain for all the research cases of BCH11 and BCH12 are the same, the angle of attack converges around the equal value for these cases. The BCH11 and BCH12 also show better tracking performance by just looking at the graphs as for these conditions, the angle of attack response is able to follow the reference signal precisely before 5 seconds. This is while, for BCH22, the angle of attack is still oscillating around the reference trajectory. Obviously, for the BCH22 case, the controller needs more exploration within the more complicated action space to find the best combination of the effectors to achieve the control task.

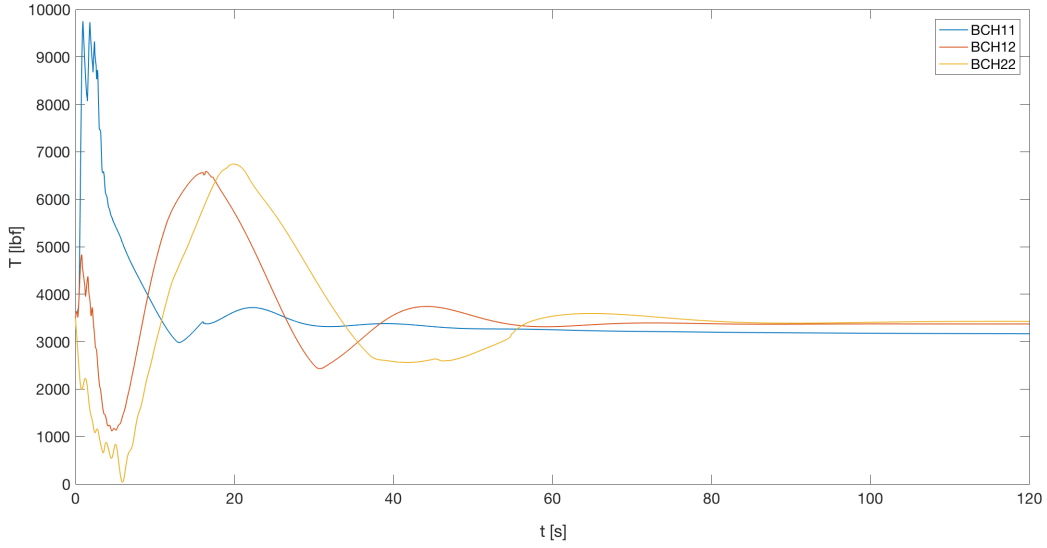


Fig. 17 The thrust values for different benchmarking conditions with the outerloop altitude control

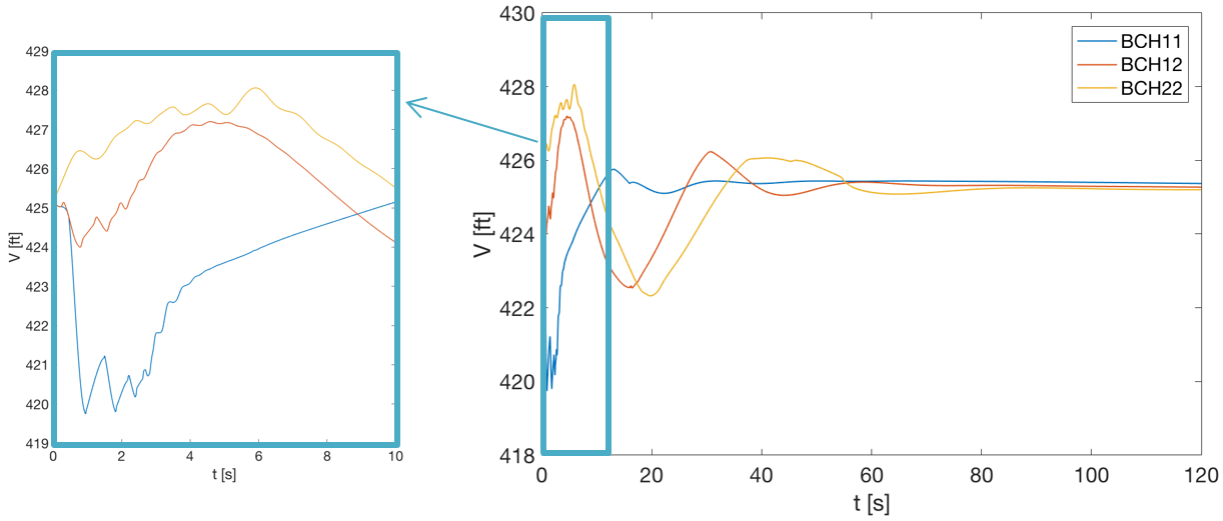


Fig. 18 The airspeeds for different benchmarking conditions with the outerloop altitude control

Although the same initial airspeed drop can be seen for cases BCH11 and BCH12, for the latter, the airspeed drop is smaller compared to the former. However, the airspeed for the rest of the learning shows some oscillations before stabilizing, which relates to the angle of attack changes. Apart from the different initial responses of the airspeed in the first few seconds, for the rest of the learning with BCH22, the airspeed tends to oscillate with the angle of attack stabilizing around its trim value. Therefore, the controller will need more than 1 minute to maintain the altitude. The gradual increase in airspeed can also be seen here for the first research case, which converges after almost 30 seconds while the altitude is still increasing towards the desired altitude.

Fig.19 shows the effectors' control inputs time histories for the controllers under two first cases. It can be seen that a different control policy is achieved for the pitch flap and elevon deflections in two cases, while there is a high-frequency usage of the control effectors at the beginning for both cases. The effectors are pushed to their limits initially, while the control policy starts becoming less aggressive sooner for the BCH12 case. For both cases, the control effector deflections utilization decrease before 5 seconds. It can be seen that the pitch flaps and elevons are used antagonistically in both cases and produce additional drag that is not desired. As already mentioned, the effort of the controllers is not included in the objective function, and therefore, there is no information provided about the effectors' deflections to the controller. Although in the attitude case, the controller is able to achieve a contributing effect from the controllers, in this case, a larger control effector utilization is achieved. However, no direct conclusion can be made by comparing the two, as there is no basis for comparison in this case. The excessive effector utilization also explains the increase in thrust for case BCH12 after 10 seconds and the decrease in the slope of the thrust behavior for case BCH11, while for both cases, the thrust values are converging to values close to their trim values.

The control effectors response for the BCH22 case are shown by Fig.20. The plot shows the same high-frequency changes in the control effectors deflections for the first few seconds. However, it can be seen that for this case, where six effectors become available for the controller, the control policy does not include any effector deflections to their saturation limits, while the deflections become smaller in frequency after 6 seconds of learning and change with a smaller slope. Another observation is that close to 18 seconds of learning, the deflections change signs and obtain larger values while the controller also starts using spoilers (SSDs). This is followed by the decrease in effectors utilization after 42 seconds and then increasing them again before 50 seconds of learning where the control policy start converging.

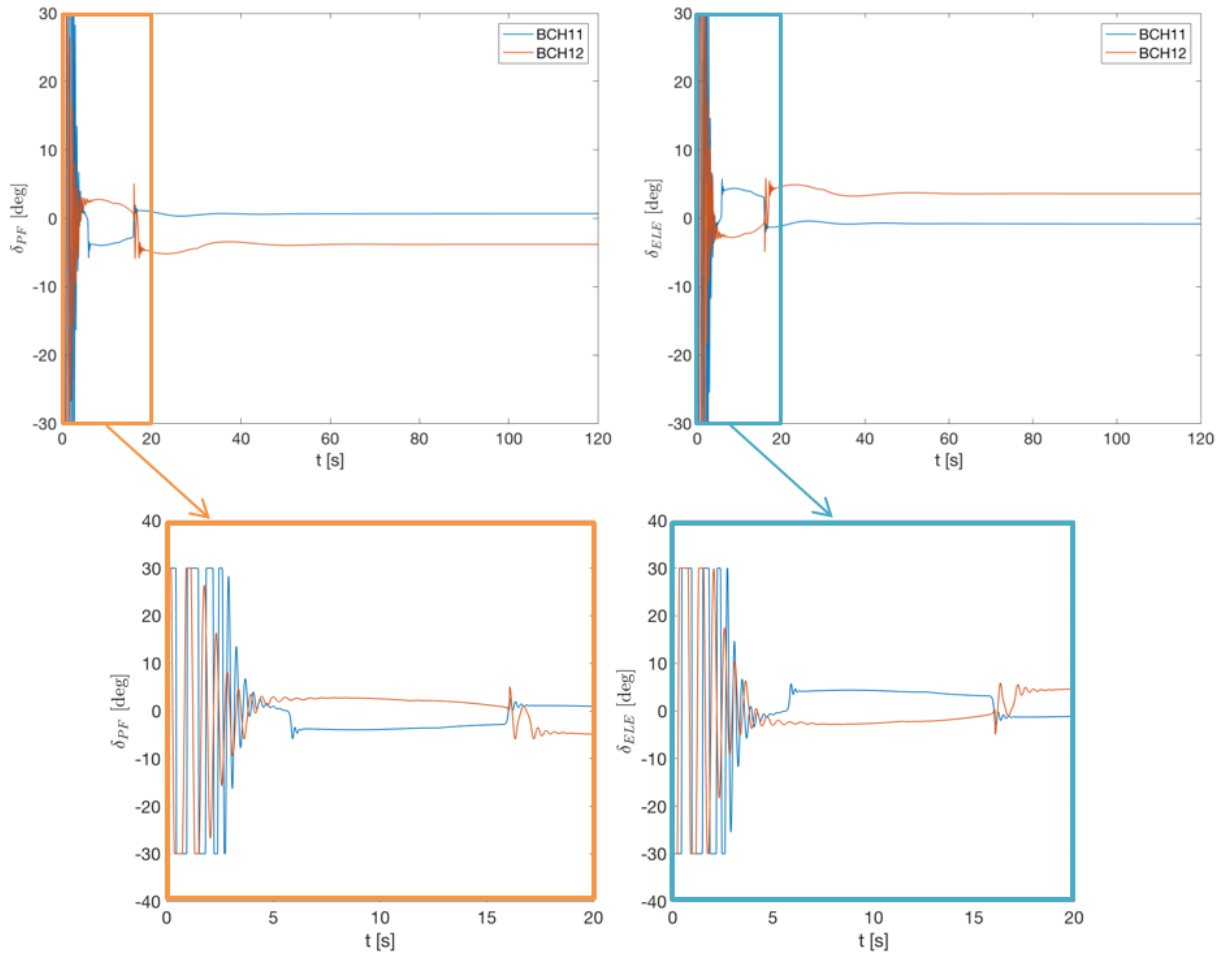


Fig. 19 The pitch flap and elevon responses for BCH11 and BCH12

It can be seen that after first high deflections of the ILEF effector, the controller decides not to use this effector for the rest of the learning after 1bout 19 seconds of online learning (due to overlapping of the AMT and ILEF graphs, it's not possible to distinguish the plots). The same can be seen for the AMT deflections, as these effectors deflections are reduced to zero by the change in the policy after 19 seconds. By just looking at the graph, one can see that the significant changes in the policy happen in 19, 45, and 58 seconds of learning before the controller convergence. It can be seen that for this case, also, the PF and ELE effectors are deflecting antagonistically, while the outboard leading edge flaps and spoilers deflections are contributing with the elevons. However, the overall effect would increase the altitude and therefore contribute to altitude regulation. This way of effector utilization, however, would result in excessive drag. Therefore, more thrust power is commanded by the aircraft after 10 seconds, as shown by Fig.17. However, the converged control policy requires almost the same thrust values that the controller converges to when compared to the two other research cases where only two effectors are used. The deflections of the rest of the control inputs remain unchanged while setting at small values.

The quantitative comparison of the different research conditions is presented by Table 8. The results presented in this table correspond to the successful runs plotted above. It can be observed that the altitude tracking error has the least value for the BCH11 case, and for the last case, the tracking error increase by almost 95 % from the first case while the angle of attack tracking error, in this case, is smaller than BCH11 and BCH12. Therefore, for the current solutions, the inclusion of more effectors resulted in a worse altitude tracking performance but a better angle of attack track. Therefore, the angle of attack tracking performance almost shows the opposite behavior than the altitude, by achieving the most tracking error for the case BCH11 and the least for BCH22. This also reflects in the Fig.16 where we see bigger amplitudes for the angle of attack oscillations around the reference signal.

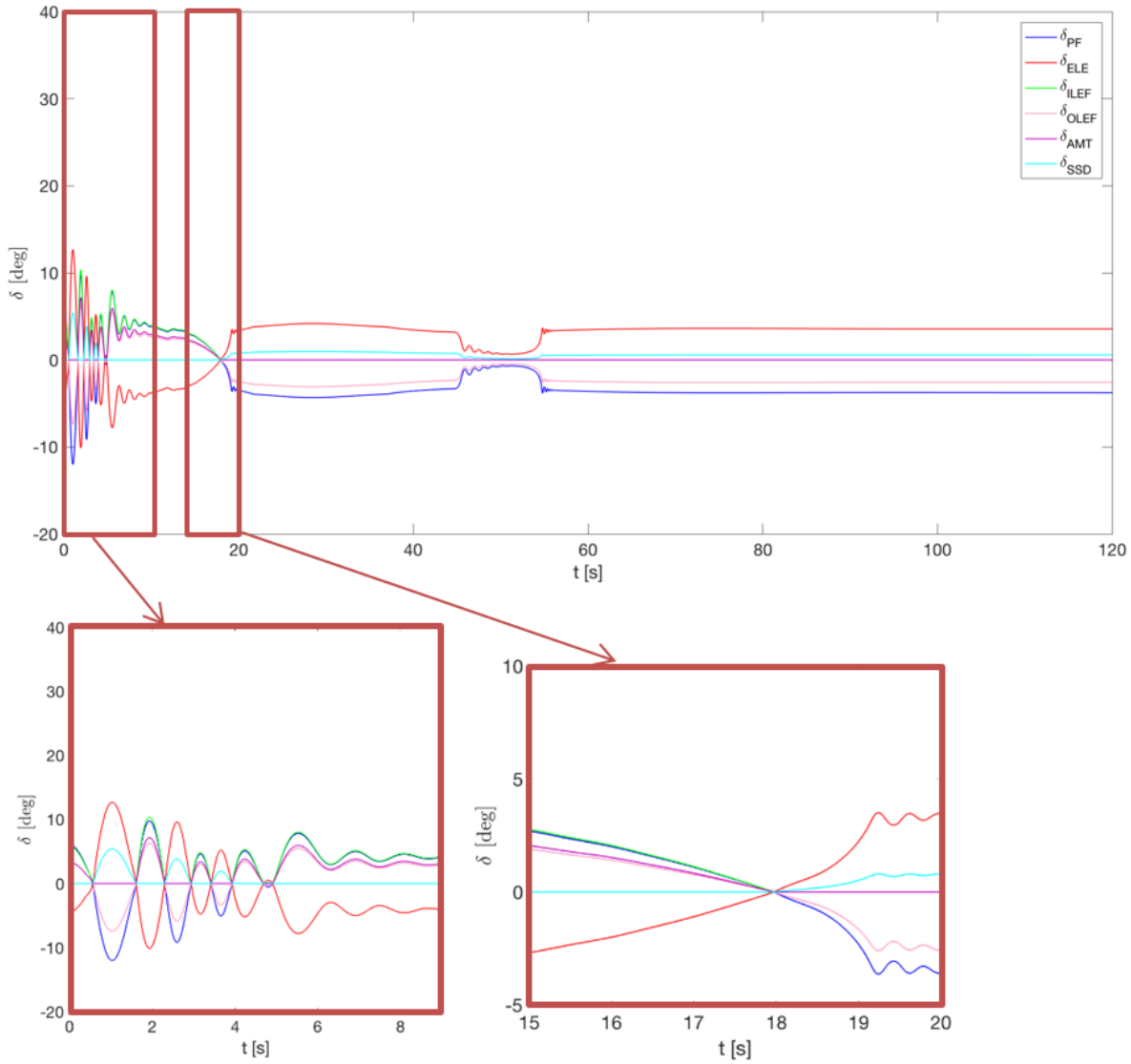


Fig. 20 The effectors responses for BCH22 case

So, the best angle of attack performance is achieved by BCH22. The results corresponding to the stability of the controller show that case BCH11 shows more than twice the success ratio when compared to the following two conditions for 100 runs of all the controllers with random initialization of the network weights. For the last two cases, the success ratio is almost the same for the two controllers.

Table 8 The altitude tracking performance for benchmark conditions

Case	RMS (z) [ft]	RMS (α) [deg]	Success Ratio [%]
BCH11	93.3719	0.4958	50
BCH12	143.2720	0.2695	21
BCH22	169.4833	0.2691	22

Table 9 shows the mean control effort for all the effectors in all the research conditions. All effector utilization (except for the MTV) is only possible for the last research case, while in the first two cases, only the pitch flap and elevons

can be used. In all instances, the cost function of the ACD controller only considers the angle of attack tracking error. In general, it can be seen that making more effectors available for the controller does not cause a lower control effort for the pitch flaps and elevons when research cases BCH11 and BCH22 are compared. Interestingly, for both effectors, the control effort is significantly smaller for the case BCH11. The control effort for the rest of the control surfaces is little, with the AMT and SSD control efforts substantially lower compared to the rest of the effectors. Therefore, the pitch flaps and elevons, which have the most direct effect on longitudinal controls are used considerably more than the effectors that are designed for lateral-directional movements.

Table 9 The altitude tracking control efforts for benchmark conditions

Case	PF	ELE	ILEF	OLEF	AMT	SSD
	MCE [-]	MCE [-]	MCE [-]	MCE [-]	MCE [-]	MCE [-]
BCH11	0.0544	0.0587	0	0	0	0
BCH12	0.1373	0.1325	0	0	0	0
BCH22	0.1178	0.1137	0.0116	0.0611	0.0056	0.0094

In the last case, the elevons control effort is almost the same as pitch flaps, and as was shown, they are moving antagonistically. This explains why to achieve the control objective; the policy contains using other effectors. As was expected, all moving tips control effectors control effort is close to zero, and they have been utilized the least. It can be seen that the OLEF has a more considerable control effort and can be seen by the Fig.20, the OLEF effectors are deflecting in contribution to pitch flaps and probably compensate for the elevons antagonistic behavior. This results directly in the decrease in the airspeed due to the increase in the drag in this duration that will be compensated by an increase in the thrust values. Introducing more effectors to the controller increase the ambiguity in the solutions, and with the effector utilization not being considered in the cost function, it can be seen that while the primary objective of altitude control is achieved, however, the effectors are not being used in an efficient way.

VII. Conclusions and Final Remarks

Adaptive control methods have been proposed for the control of the automated systems. Most of these approaches are model-based or cannot be applied online. The Innovative Control Effector (ICE) aircraft design comes with a large control suite that results in very complex dynamics for the aircraft. Therefore, a model-free control approach that is adaptive and can deal with the complicated system of the ICE aircraft is beneficial. A controller design based on continuous Reinforcement Learning (RL) algorithms can provide an adaptive controller for the ICE aircraft that is model-free and can be used for online control of the aircraft.

The results for the application of the approximate actor-critic-based reinforcement learning method of Heuristic Dynamic Programming (HDP) for control of the nonlinear innovative control effector aircraft model using HDP shows the advantage of this approach in two ways. First, the ability of the approximate RL controller is shown in achieving an attitude tracking task with online internal local model identification. Second, the controller is able to learn online and without prior knowledge of the system dynamics, to control the longitudinal dynamics of the aircraft in achieving the desired altitude by integrating the RL controller with an outer loop PID controller, both learning online and simultaneously. The results are shown for symmetric utilization of different configuration of the control effectors. In all cases, the tracking error is the only source of information for the controller. While the HDP controller is successful in both problems of attitude control and altitude control of the ICE aircraft, it is concluded that achieving the control objective, especially for the second problem, results in inefficient effector utilization, causing excessive drag and, therefore, misuse of thrust power. This means that for better and more efficient use of the effectors, the control effector utilization and possibly thrust usage should be directly included in the cost function. It is shown that inefficient control effector utilization is not entirely reflected in the attitude control problem or when small numbers of effectors are considered. However, for altitude control and with the introduction of the new effectors, this issue becomes more important for the controller.

The quantitative comparison for attitude control with a different number of effectors shows that one of the best tracking performances are obtained when the controller considers pitch flaps and elevons as the available control powers.

It is also concluded that the increase in the number of effectors results in longer run times and adding the effectors of leading-edge flaps, all moving tips and spoilers reduces the success ratio significantly. In general, it is found that increasing the number of effectors does not necessarily decrease the control effort in attitude control, and some cases increase the effort for effectors that are designed for longitudinal control, like pitch flaps and elevons). For the altitude control case, it is shown that while the solution depends on the gains of the outer-loop controller, the altitude error increase with adding more effectors while the angle of attack error decreases. Also, a more success ratio is achieved if the altitude control is performed with less overshoot. In this case, too, adding more effectors is shown not to decrease the control effort of the controllers necessarily.

Finally, for future studies, the RL controller implementation for higher order control problem of altitude control with no cooperation with the PID controller should be investigated. For the current proposed framework, the effectors' response to the inclusion of effector utilization in the cost function can be considered to solve the problem of inefficient effector use and as a result, to avoid the excessive drag. One downside of the HDP control with neural networks, which is inherited by policy gradient methods, is the probability that you might trap in a local optimum for some function approximation approaches. One solution can be to use Softmax functions or by using TD prediction methods (e.g., using target networks). Finally, one of the vital control power available for longitudinal control of the ICE aircraft is the thrust vectoring, or more precisely, the pitch thrust vectoring. In future studies, this effector should also be considered for the investigations with RL using the complete control suite.

.

References

[1] Airplanes, C., "Statistical summary of commercial jet airplane accidents," *Worldwide Operations*, Vol. 2008, 1959.

[2] Dorsett, K. M., and Mehl, D. R., "Innovative control effectors (ICE)," Tech. rep., Lockheed Martin Tactical Aircraft Systems Fort Worth Tx, 1996.

[3] Bowlus, J., Multhopp, D., Banda, S., Bowlus, J., Multhopp, D., and Banda, S., "Challenges and opportunities in tailless aircraft stability and control," *Guidance, Navigation, and Control Conference*, 1997, p. 3830.

[4] Niestroy, M. A., Dorsett, K. M., and Markstein, K., "A Tailless Fighter Aircraft Model for Control-Related Research and Development," *AIAA Modeling and Simulation Technologies Conference*, 2017, p. 1757.

[5] Whitford, R., "Design for air combat," *RUSI Journal*, Vol. 132, 1987, pp. 79–80.

[6] Thrun, S., and Littman, M. L., "Reinforcement learning: an introduction," *AI Magazine*, Vol. 21, No. 1, 2000, pp. 103–103.

[7] Zhou, Y., Van Kampen, E., and Chu, Q. P., "Incremental approximate dynamic programming for nonlinear flight control design," *Proceedings of the 3rd CEAS EuroGNC: Specialist Conference on Guidance, Navigation and Control, Toulouse, France, 13-15 April 2015*, 2015.

[8] Zhou, Y., van Kampen, E.-J., and Chu, Q. P., "Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming," 2017.

[9] Zhou, Y., van Kampen, E.-J., and Chu, Q. P., "Incremental model based online dual heuristic programming for nonlinear adaptive control," *Control Engineering Practice*, Vol. 73, 2018, pp. 13–25.

[10] de Vries, P. S., "Reinforcement Learning-based Control Allocation for the ICE Aircraft Model," Tech. rep., Delft University of Technology, 2017.

[11] Ferrari, S., and Stengel, R. F., "Online adaptive critic flight control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 777–786.

[12] Ferrari, S., Steck, J. E., and Chandramohan, R., "Adaptive feedback control by constrained approximate dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 38, No. 4, 2008, pp. 982–987.

[13] Lewis, F. L., Vrabie, D., and Syrmos, V. L., *Optimal control*, John Wiley & Sons, 2012.

[14] Zhou, Y., "Online reinforcement learning control for aerospace systems," Ph.D. thesis, Delft University of Technology, 2018.

[15] van Kampen, E.-J., Chu, Q., and Mulder, J., "Continuous adaptive critic flight control aided with approximated plant dynamics," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006, p. 6429.

[16] Powell, W. B., "What you should know about approximate dynamic programming," *Naval Research Logistics (NRL)*, Vol. 56, No. 3, 2009, pp. 239–249.

[17] Bradtke, S. J., and Barto, A. G., "Linear least-squares algorithms for temporal difference learning," *Machine learning*, Vol. 22, No. 1-3, 1996, pp. 33–57.

[18] Eerland, W., de Visser, C., and van Kampen, E.-J., "On Approximate Dynamic Programming with Multivariate Splines for Adaptive Control," *arXiv preprint arXiv:1606.09383*, 2016.

[19] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction "Complete Draft"*, MIT press Cambridge, 2018.

[20] Modares, H., Sistani, M.-B. N., and Lewis, F. L., "A policy iteration approach to online optimal control of continuous-time constrained-input systems," *ISA transactions*, Vol. 52, No. 5, 2013, pp. 611–621.

[21] Si, J., Barto, A. G., Powell, W. B., and Wunsch, D., *Handbook of learning and approximate dynamic programming*, Vol. 2, John Wiley & Sons, 2004.

[22] Kiumarsi, B., Vamvoudakis, K. G., Modares, H., and Lewis, F. L., "Optimal and autonomous control using reinforcement learning: A survey," *IEEE transactions on neural networks and learning systems*, Vol. 29, No. 6, 2018, pp. 2042–2062.

[23] Modares, H., Lewis, F. L., and Naghibi-Sistani, M.-B., "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, Vol. 50, No. 1, 2014, pp. 193–202.

[24] Dierks, T., and Jagannathan, S., "Online optimal control of nonlinear discrete-time systems using approximate dynamic programming," *Journal of Control Theory and Applications*, Vol. 9, No. 3, 2011, pp. 361–369.

[25] Luo, X., and Si, J., "Stability of direct heuristic dynamic programming for nonlinear tracking control using PID neural network," *Neural Networks (IJCNN), The 2013 International Joint Conference on*, IEEE, 2013, pp. 1–7.

[26] Buffington, J., and Buffington, J., "Tailless aircraft control allocation," *Guidance, Navigation, and Control Conference*, 1997, p. 3605.

[27] van der Peijl, I. V., "Physical Splines for Aerodynamic Modelling of Innovative Control Effectors," Master's thesis, Delft University of Technology, 2017.

II

Literature Review and Preliminary Analysis

2

Reinforcement Learning

In this chapter, a brief but comprehensive overview of Reinforcement Learning (RL), as one of the main theory contents, which are referred to in the thesis, is studied. The focus of the investigation is on Continuous RL methods and the literature available in approaching RL control problems in continuous state and action spaces. RL can be assumed as an action-based and highly goal-oriented learning approach among other methods of machine learning, e.g. supervised and unsupervised learning. This chapter starts with an introduction to RL in section 2.1. Reinforcement problem elements and solutions along with other terms associated with RL are given in section 2.2 and section 2.3, respectively. The classification of RL to discrete and continuous methods are illustrated in section 2.4. section 2.5 includes the introduction of ACD methods as one of the promising structures for approximate RL control problems. Safety in RL followed by Lyapunov- based RL is described in section 2.6. Finally, section 2.7 gives a conclusion on the chapter. Some of the materials presented in this chapter are paraphrased from the book (Sutton & Barto, 1998). It should be noted that discrete and continuous terms in this study correspond to discrete and continuous action and space and not necessarily discrete or continuous-time calculations. In this chapter, the RL problem is mostly looked at from an artificial intelligence perspective and therefore, some terms are used interchangeably with the control domain.

2.1. Introduction

Reinforcement Learning is the idea of active decision-making and learning by interacting with the environment. The idea of RL is inspired by human and animals learning while being a logical-mathematical foundation. But why RL, between other machine learning methods, stand out? The question can be best answered by comparing RL with other learning methods. Although RL and supervised learning have some common grounds, the learning procedures in these approaches are different. For example, at the beginning of the 1980s, most of the work done by Sutton and Barto (1981) was to emphasize such a difference. Inductive supervised learning methods, which are learning methods for an observable environment, have reached high levels of complexity handling. When a chaos date is given with some information on the nature of the data, supervised algorithms are able to extract features of the data and categorize them by operating an optimization procedure on a defined error criterion. So, Supervised learning is a data classifier based on the knowledge provided by an external designer. In supervised learning, the correct control output to the controller is known, and the learning occurs based on the error corresponding to the difference between the correct control output and the controller output (learning with targets). In some cases, supervised learning are used in parallel with RL (Ferrari and Stengel, 2004).

However, in learning to control, especially for more complex dynamics, the task usually is not well defined. In other words, the necessary control actions are not always available. Therefore, in these problems, the goal is to learn the best actions in response to a perceived favourable state to make the system accomplish the given task (or reach the favourable states), e.g. stabilization in a target state or reaching a certain state (van Kampen *et al.*, 2006). The general structure for reinforced problems is

described in terms of optimization of a performance measure with no pre-determined data fed to the training, which is the case for supervised learning. Therefore, for an RL setting, the task will be learned from scratch and by assigning credits (abstract reward/cost signals) to the actions based on their long-term influence. This result in need for dense exploration which would cover a significant fraction of the entire state and action space which would result typically in computationally intensive algorithms. In some approaches, learning is benefited by suitable transfer of the information about the previously learned tasks or even by an expert knowledge (learning from demonstration) (Schaal, 1997). Still, the exploration-exploitation dilemma is the subject of study for mathematicians for many decades. But, learn based on RL agent (controller) own experience and despite uncertainty about the environment, is the main attractive feature of RL. Within RL algorithms, the ones dealing with continuous action and state spaces have the challenge of reliability and consistency, which become highlighted for complex systems. In the following sections, the RL problem is described in more details.

2.1.1. General description

RL methods only need the data that is obtained from the system and are independent of a behavioral model of the system. That is the most important feature of these algorithms. Being a Machine Learning (ML) approach, RL is focused on goal-directed learning, which can use its experience to improve its performance over time. Maximizing a numerical reward signal by mapping situations (states) to exploratory actions (transitions) is the working principle of the RL. The most two important features of RL are trial-and-error and delayed rewards. RL evaluates the actions taken by the agent (controller) using training information instead of instructing them with the best action. This feature makes RL distinguishably different from other types of learning.

2.1.2. Background on reinforcement learning

Reinforcement Learning (RL) became attractive to scientists by the late 80s when computer powers became compatible with RL applications. What is known as modern RL introduced by the late 1980s when two separate threats of optimal control by dynamic programming solutions and trial and error from artificial intelligence came together. Trial and error learning was first computationally studied in the work of Farely and Clark (1954), which included simulating artificial neural networks. Later they switched their focus on supervised learning and pattern recognition. However, the term ”reinforcement learning” was only used by literature in the 1960s, with the most highlighted paper of Minsky (1961), with discussions on reward and penalty assignments known as *credit assignment problem*. Some of the highlights of works on reinforcement learning which influenced the path of its development are mentioned in the following.

The work by Michie (1963) on playing tic-tac-toe using trial-and-error learning with reward and punishment setting and Michie and Chambers (1968) on using a reinforcement learning controller for balancing a pendulum attached to a movable car based on a failure signal, are the most distinguished studies on RL before the 80s. These studies are the basis for the work of Barto *et al.* (1983) followed by Sutton (1984). Using RL for the balancing task of a pendulum is one of the early works which showed the use of RL with systems without complete knowledge. Before that, the same task was studied by Widrow and Smith (1964) with supervised learning and with using expert knowledge instruction. Later, Widrow *et al.* (1973) used a modified adaptive filter and introduce a form of learning ”learning from a critic” which was based on ”selective bootstrap adaptation” and was compared to supervised learning which used training sets for learning.

What lead to the studies in the modern reinforcement learning is originated in the Soviet Union by Tsetlin (1973) and started from researches on ”learning automaton.” These methods, known as *n-armed bandit*, are simple and low-memory based and are purely based on choices. Later by the work ofHolland (1975), the theory of adaptive systems based on trial and error selection and including association with value functions was introduced.

Another important associative to the modern RL is temporal-difference (TD) learning, which uses as an evaluation tool to improve learning. TD introduced in trial-error learning by Klopf (1972). He was inter-

ested in learning for large systems and therefore introduced generalization to reinforcement learning. Klopf research was followed by Sutton and Barto (1981). In this study, the TD learning and trial-and-error learning came together as the *actor-critic* structure (AC). This method was applied for pendulum on the cart problem in Barto *et al.* (1983). Later back-propagation neural networks was added to this structure in Anderson (1989).

The TD methods, optimal control, and trial-and-error learning all joined finally by the introduction of Q-learning by Watkins and Dayan (1992). Werbos (1987) contributed to this integration by studying the convergence of dynamic programming with trial-and-error learning. One of the most successful implementations is TD Gammon of Gerry Tesauro, a backgammon playing program.

Therefore, reinforcement learning is still a growing field, and most of the research problems are still open for further studies. What discussed in this section was a history of what is known as modern reinforcement learning. For more study on the history of the development of the RL methods in three aspects of state-space dimensionality, exploration/exploitation dimensionality, and value estimation approaches, the reader is referred to Ravishankar and Vijayakumar (2017). In the following, the modern developments in RL is discussed separately, focusing on methods deal with continuous action and spaces.

2.2. Reinforcement Learning Problem

The RL problem is the interaction between the decision-maker called the *agent* and what the agent is interacting with, which is called the *environment*. A simple diagram of such an interaction is depicted in Figure 2.1. At each time step, the agent receives the environmental state $s(t)$ and base on the possible states space S, select an action $a(t)$ from a set of actions $A(s(t))$ (if discrete) available at state $s(t)$. At a one-time step later, according to the action selected, a reward will be assigned for the agent at the instance it reaches the new state $s(t+1)$. Here, the policy $\pi(s,a,t)$ is the mapping from states to probabilities of selecting each possible action. RL involves the change of the agent's policy based on its experience to maximize/minimize long-term rewards/costs. In some cases, the boundaries for the learning agent is then defined at its control limits.

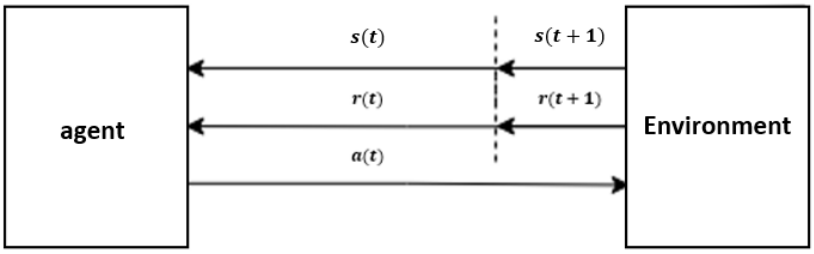


Figure (2.1) A schematic diagram of the agent and environment interaction.

In most of the RL problems a state is measured and an action corresponding to that state is chosen, a transition to the next state is measured, and the value function (and/or the policy function) will be updated. In the following, the decision making process for discrete-time systems with the framework of Markov decision processes (MDPs) is discussed; nevertheless, not all problems fit in an MDP structure. More, the elements of the RL problem along with other terms associated with are described.

2.2.1. Markov decision process

The cornerstone of all RL methods with more than one state (i.e., not a bandit problem) is the Markov Decision Process (MDP). In case, a problem is modeled as MDPs, RL algorithms can be applied upon it. An MDP can be formulated as a tuple (X, A, f, V, γ), where X corresponds to the state space, A is the action space, $f : X \times A \times X \to \mathbb{R}_+$ is the transition probability density function between the states which describe the stochastic process to be controlled. $V : X \times A \times X \to \mathbb{R}$ is the long-term reward function or

value function and γ is the discount factor in case of a discounted reward function (e.g., not average reward). The elements of the MDP can be stationary or non-stationary, depending on whether they change over time. In the case of a continuous state-space, the probability will be defined for a certain state region since reaching a particular state probability equals to zero. The probability of reaching the state $x(t+1)$ at time $t+1$ from state $x(t)$ in the $x(t+1) \subset X$ after applying control action $a(t) \subset A$ is defined as:

$$P(x(t+1)), X|x(t), a(t)) = \int_X f(x(t), a(t), x')dx' \tag{2.1}$$

The immediate reward for each state transition usually depends on the previous state, the transition state and the action taken is given by:

$$r(t+1) = V(x(t), a(t), x(t+1)), \tag{2.2}$$

The action at each state is given by the policy function $\pi : X \times A \mapsto R_+$. The RL agent aims to find the policy function π, which would maximize the expected value of the long-term reward function following the policy π. The cost-to-go function is the expected value and is given by:

$$J(\pi) = E\{g(r_1, r_2, ...)|\pi\} \tag{2.3}$$

The cost-to-go function is usually defined by two main settings of the discounted sum of rewards or the average of the immediate rewards received (Grondman, 2015).

2.2.2. Reinforcement learning elements

Any reinforcement learning approach has function components that construct the RL problem. In the following the four main elements of an RL system which are shared between all such problems are presented with a brief definition for each:

Policy:

the agent's way of behaving at a given time. The policy $\pi : S \times A \rightarrow [0, 1]$ can be a simple lookup table or and extensive computation such as a search process and is given by:

$$\pi(a|s) = P(a(t) = a|s(t) = s) \tag{2.4}$$

Policies can be stochastic or deterministic, probabilistic, or non-probabilistic. In discrete action spaces, policies can be $\epsilon - greedy$, where $\epsilon \in [0, 1]$. For a greedy policy, the probability that the agent would choose the action that would result in the highest value function equals ϵ.

Reward function:

the goal in a reinforcement learning problem. It defines good and bad for the agent. This function must be unalterable by the agent, but it is the basis for altering the policy, and it can be stochastic. Rewards can be seen as performance feedback for the controller. The sequence of rewards, in the long run, is called the expected *Return*. The return can be defined as a simple sum of the rewards for episodic tasks with specified terminal states or an infinite sum of returns with a *discount rate* for continuing tasks (considering discounted reward). The unifying term for the return can then be defined as:

$$R_t = \sum_{k=0}^{T} \gamma^k r(t+1) \tag{2.5}$$

in which T is the final time step with $T =$ corresponding to continuing tasks, γ to the discounting factor, and r is the immediate reward. As γ approaches 1, future rewards are considered by the agent more strongly. The notion reward function can also be in terms of punishments to the agent and therefore form a cost function.

Value function (utility function):

specification of the "good" behavior in the long term. Whereas rewards determine the immediate desirability of the environmental states, value functions are the indication of the long-term desirability of the state after taking into account the possibility of the happening states and the rewards available for them. The RL seek actions that bring about states with the highest value, not the highest rewards (or the lowest costs). However, it is much more difficult to determine values that it is to determine rewards. Rewards are directly given by the environment, while values need to be estimated and from the sequences of observations over the agent lifetime (Sutton and Barto, 1998). The most crucial component of all approximate RL algorithms is the method for efficient estimation of the value function. Value functions can be state-value function, denoted by V, or action-based value functions denoted by Q-functions, which determine the quality of the state and action combinations. A general definition of value functions and Q-functions can be written as:

$$V^{\pi}(s) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^{k} r(t+1) | s(t) = s\} \tag{2.6}$$

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^{k} r(t+1) | s(t) = s, a(t) = a\} \tag{2.7}$$

Q-functions are usually preferred since it is easier to find greedy policies in this approach (Buşoniu *et al.*, 2011). In policy improvement with Q-functions, a model of the environment transition dynamics is not required. In some literature, advantage function is defined and estimated, which is the difference between Q-function value and the value function estimate: $A^{\pi}(x,a,t) = Q^{\pi}(x,a,t) - V^{\pi}(x,t)$. The value of this function gives a relative measure of the value of each action since its expectation is zero for each time step. In summary, the value function is a memory structure for storing the relation between behavior and rewards, which allows the agent to make decisions for future actions based on its current state and past experiences.

Model (of the environment):

The RL method uses trial and error to learn a model of the environment and uses the model for planning. The planning can go from low-level trial and error learning to high-level, deliberative planning. The model for an RL agent can be implicitly stored in the value function or identified (will be discussed by section 2.5), and therefore, RL is a model-free control approach.

Learning parameters:

Some other properties, known as learning properties, defined for RL agent training, are as follow:

- **discount factor (γ):** a trade-off indication between immediate and future rewards.
- **learning rate (α):** determines the amount the newly acquired data override the previously obtained information.
- **eligibility traces (λ):** for better assignment of credits to state-action pairs that have been visited several times during learning, eligibility traces are used. This means that the recently visited states or state-actions will become more eligible for receiving credits. Eligibility traces have shown to increase the speed of learning (Sutton, 1988).

2.3. Reinforcement Learning Solutions

In this section, the fundamental methods for solving finite MDP problems are described. The pros and cons of each technique are discussed, along with their learning performances. Although most of these methods are categorized as tabular methods by Sutton and Barto (2018), it is beneficial to give a brief introduction to them. In this section, different RL solution methods are defined only based on their structures and not based on their approach in solving various components of the problem, e.g., if discrete or continuous state and action spaces are considered. Such topics are discussed in section 2.4.

2.3.1. Dynamic programming

The basis of each Dynamic Programming (DP) algorithm is the Policy Evaluation (PE) or prediction and Policy Improvement (PI). The parallel repetition of PE and PI is called policy iteration that will result in the optimal policy. During learning, the agent will have to estimate the cost-to-go function J for a given policy π. This procedure is called policy evaluation. The resulting estimate of J would be the value function. An advantage of DP algorithms is that the PE and PI can be applied at each time step and do not need a full episode to finish. DP methods are well developed mathematically, but they depend on the accurate global model of the environment.

2.3.2. Monte Carlo methods

Monte Carlo (MC) methods, unlike DP methods, learn the optimal policy based on episodic steps. Therefore, no global modeling is required for this approach. However, these methods accuracy will be deteriorated for a large number of states. An advantage of MC is its flexibility for problems that might not completely follow MDP properties. MC methods follow the scheme of policy iteration by deriving the value of each state with averaging returns that start from that state. Therefore, the expected return will be the value of the state. In the form of sample episodes, MC methods do not require bootstrap; that is, policy evaluation does not include updating the estimation of the value function based on other value estimates.

2.3.3. Temporal-difference learning

Temporal-Difference (TD) learning is a model-free based RL algorithm, which learns the values with trial-error and is one of the novel ideas in RL. It can be seen as integration between MC and DP methods, in a sense that is model-free as MC and like DP it bootstraps. While TD and DP methods are quite similar in the sense that they both try to make each state consistent with its successors by making local adjustments to the value function estimates, there are still some major differences. First, TD has a backward approach in adjusting the states by looking at all observed successors of the state. DP, however, adjust the states based on the weighted probability of all the probable successors of the current state. However, this effect tends to disappear as TD adjustments are averaged (Russell and Norvig, 2016). TD methods convergence might be slower and with higher variabilities when compared to DP methods, but the implementation is much simpler, and it also requires less computation per observation, since a transition model is not required to perform the updates. The update rule in a TD algorithm for the value function is given as:

$$V(s,t) \leftarrow V(s,t) + \alpha[r(t+1) + \gamma V(s,t+1)) - V(s,t))] \tag{2.8}$$

where $r(t+1) + \gamma V(s,t+1)$ is the target for updating the value function estimate and if $TD(0)$ is considered. TD methods application for solving control problems is abstracted from DP algorithms. They can be classified based on their exploration maintenance into off-policy and on-policy methods. There are generally three ways that TD methods can be extended to control problems, the Q-learning, Sarsa, and actor-critic methods. Before going to describe each method, first, off-policy and on-policy methods are described in the next part.

Off-policy vs on-policy

RL methods can be separated into two classes based on the way the algorithm reaches the goal, so he off-policy and on-policy algorithms. Off-policy methods are normally used in value-iteration or critic-only control problems and include a policy that generates the behavior of the agent, called behavioral policy and a policy that is used for PE and PI steps. These two policies do not relate to each other. In this approach, the optimum Q-function is found and then will be used to find the optimum policy (or action), by solving the Bellman optimality equation. Hence, Q-value will be updated at each state and action by choosing for the maximum value of Q-function, i.e., using the Q value for the next state and the greedy action. The off-policy methods can be performed both for offline batch data and in online mode. They are considered more data-efficient and faster in learning since the experiences learned from executing behavioral policy can be reused for updating the value-function of different target policies, although they might lack enough exploration. These methods are studied for different approximation methods, such as fuzzy rule-based in the work of Glorennec (2000) and with neural networks by Lin (1992) (These are not the only authors). The extension of off-policy methods to approximation is a cutting-edge research area, and the developed theoretical results are not strong enough to be discussed here.

In on-policy methods, on the other hand, the policy remains unchanged in updating the Q-function. Policy iteration methods are categorized as on-policy methods. The difference between on-policy and off-policy methods lies in the fact that in on-policy methods, the value-function is estimated for a policy currently in use while in off-policy case the current policy, the behavior policy, is unrelated to the policy that is being evaluated improved (the estimation policy). Policy iteration methods normally start with a stabilizing control policy and solve Bellman equation until the optimal control policy is obtained, while in value-iteration algorithms, an admissible control policy is not necessarily the initialization point for iterations (Kiumarsi *et al.*, 2018). The downside with on-policy methods is that there is no proof of convergence for them to near optimality (Grondman, 2015), while divergence is shown in some cases (Melo *et al.*, 2008). However, convergence for linear-in-parameters function approximations with some constraints on the sampling trajectories is shown. Although conditions for convergence and bounding the approximation errors for linear FAs are studied inSchoknecht (2003), it is shown that learning value functions by TD methods will be biased and the bad choice of the basis functions can result in even large biases (although these methods mostly are focused on linear function approximations).

There can also be a third categories of policy search (actor-only methods) which look directly for the optimum policy, by finding the greedy policy in optimal value-function or by performing optimization procedures on parameterised policies (Grondman (2015), Konda and Tsitsiklis (2003), Berenji and Vengerov (2003)). Therefore, they can generate and handle continuous action. However, the learning rate for actor-only methods is slow since they suffer from high variance in the gradient estimate (Grondman, 2015). Online planning is proposed for this approach by Buşoniu *et al.* (2016). The main advantages of the actor-only methods are their inherited convergence obtained by gradient descent methods and their ability to handle continuous actions. The drawback is the large variance of the estimated gradient and that they do not include past experiences (Riedmiller *et al.*, 2007).

Off-policy Q-Learning

A TD learning approach, introduced by Watkins (1989), that learn the action-based value function in the training phase, comparing to TD which learn the state-based value function. The Q-learning algorithms do not need a model of the system in learning and action selection phases, e.g., in the form of a probability function (Russell and Norvig, 2016). Therefore, an algorithm is needed that would estimate and learn the expected long-term rewards. This is what Q-Learning is designed to provide. Q-learning (Bradtke *et al.* (1994), Watkins and Dayan (1992), Watkins (1989)) , can interpret the expected values for its available action choices, independent of the their outcomes (a characteristics of model-free methods). However, the learning ability in Q-learning is limited by their lack of predictions of these action choices (i.e., not knowing where the actions lead (Russell and Norvig, 2016)). For the TD Q-learning method, the update of the value-function occurs as:

$$Q(s,a,t) = Q(s,a,t) + \alpha(R(s,t) + \gamma \max_{a(t+1)} Q(s,a,t+1) - Q(s,a,t)) \tag{2.9}$$

The model-free nature of the Q-Learning reflects an iterative estimation of the value function, without estimating the Markov Decision Process. Also, the learning of value function and policy happens simultaneously with Q-learning. One downside is that the off-policy methods are biased while being sample efficient (Islam and Seraj, 2017) and, therefore, usually have convergence issues.

On-policy SARSA

Introduced by Rummery and Niranjan (1994) SARSA, stands for State-Action-Reward-State-Action, SARSA is a model-free policy iteration method. It can be assumed as the close relative to Q-learning if the agent is purely greedy, meaning that the agent always picks the best action with the highest value function. Q-learning differs from SARSA if exploration is applied for the agent. Q-learning will always choose the highest Q-values at each state reached, while SARSA waits for the action to be taken and then support the Q-value for the selected action. The update equation of the SARSA method for value-functions is as:

$$Q(s,a,t) = Q(s,a,t) + \alpha[R(s,t) + \gamma(Q(s,a,t+1) - Q(s,a,t)] \tag{2.10}$$

Russell and Norvig (2016) sees Q-learning as a more flexible approach, able to keep good behavior even in the presence of a random or an exploration policy. However, he defines SARSA as a more realistic approach, since it shows the true behavior of the environment instead of the way the agent would like the environment to behave. In contrast to Q-learning method, for SARSA, the estimate of the value function depends on the exploration policy, considering it be an on-policy approach. It is possible for a SARSA algorithm to stuck in the local minimum. Although on-policy methods result in stable learning, however, a drawback would be the cost of sample complexity (Islam and Seraj, 2017). A modification on SARSA algorithm called Expected SARSA is given by Sutton and Barto (2018). This method is considered off-policy and follows the same schema as Q-learning but moving in the same direction as SARSA and therefore, is called Expected SARSA. In this approach, instead of the Q-learning maximum over the next Q function pairs, the expected value, which is the measure of how likely each action is under the current policy, is used. The advantage of the approach over SARSA is that the variances that are caused by random selection between available actions are eliminated.

Actor-Critic methods

A popular policy gradient method that is useful for control systems with continuous states and actions and, therefore, for real-world applications and problems is the Actor-Critic (AC) structure. Policy gradient methods are a group of RL algorithms that rely upon gradient descent methods to optimize parametrized policies with respect to the long-term cumulative reward/cost or the expected return. In TD methods, agent evaluates a particular policy, so learning of policies occur. In AC methods, however, the policy changes in learning of the value function. In this structure, the purpose is to have a separate presentation of the policy from the value-function. Therefore, it can be assumed as a general structure of RL implementations, which makes a division between the task or value-function estimate (critic) and the policy (actor). In such an architecture, the actor proposes the subsequent actions by interacting with its environment, while the critic evaluates the actions and provides the actor with the feedback and improves the performance of the next actions of the actor. AC methods use function approximation since they are dealing with continuous platforms. A more detailed description is given in section 2.4. A schematic for the actor-critic network is as depicted in the Figure 2.2 for discounted return.

Actor-critic methods combine learning in policy space with value function approximations result in algorithms which prove convergence for certain time-scales (Konda and Tsitsiklis, 2000). Therefore, they do not suffer from the complexity problem which arises for most of the traditional RL problems in continuous state and action spaces and the intractability problem caused by uncertainty in the state information. Also, in this approach, the lack of guarantees for convergence of the value function is eliminated. However, the convergence of the actor-critic methods is still highly dependent on the initial

policy. Besides, the learning rate for the policy for an AC architecture is usually lower than methods that only include value function iterations, and their convergence depends on the convergence of the two-time scale that the stochastic algorithms used for estimating the actor and the critic.

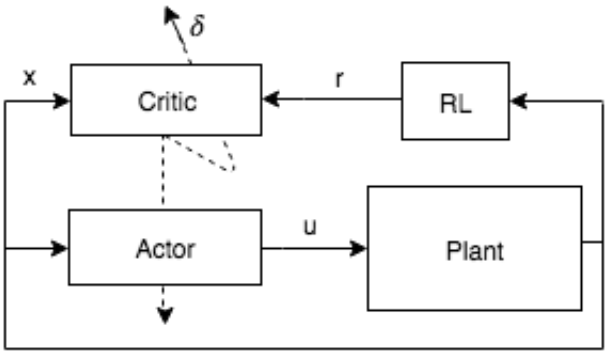


Figure (2.2) Schematic of the actor-critic structure

A classification based on the actor-critic structure for RL methods can be critic-only, actor-only, and actor-critic designs. Q-learning and SARSA would be placed in critic-only methods, in which generating a control action require optimization procedure over the value function (Grondman, 2015). These methods use the approximation or discrete Q-function for solving the RL problem without using an explicit policy function, and then the optimal value function is obtained by finding an approximate solution to the Bellman equation online. In an online setting, however, there is no guarantee for the policy to be near-optimal and it is highly dependent on the estimation of the value function (Baird (1995), Gordon (1995), Tsitsiklis and Van Roy (1996)). In some cases, a divergence of the approximation is shown (Melo *et al.*, 2008) and that the convergence can be shown for approximations that are linear-in-parameters and if the trajectories are sampled in an on-policy manner. In general, critic-only methods normally suffer from convergence problems (Grondman, 2015).

Actor-only methods such as REINFORCE algorithms (Williams, 1992), also known as Monte-Carlo policy gradient methods, do not used the stored knowledge of the value-functions and need large on-policy samples (Islam and Seraj, 2017). In most cases, they used an approximation of policies and optimize the cost-to-go function determined under policy space in the parametric case. One major advantage of these methods, comparing to critic-only methods, is that they can generate policies in all over continuous action spaces (Grondman, 2015). The convergence for actor-critic methods is naturally inherited from their gradient-descent property (Sutton *et al.*, 2000). However, these approaches tend to forget their past experiences (Konda and Tsitsiklis (2003), Peters *et al.* (2010)) and it should be noted that two convergences are important in evaluating AC methods, the convergence of value function and the policy, with the, emphasize on the latter.

Actor-critic, usually considered as an off-policy method, combine the critic-only and actor-only methods to benefit the advantages of both. However, in some approaches, AC is used to combine off-policy and on-policy methods, which would only be possible in an AC framework. In this architecture, the critic approximates the value function using the available samples (policy evaluation) and then use the value function to update policy parameters in actor to improve the performance (policy improvement). In addition, they enable the use of continuous actions by getting the policy directly from the approximated policy function. Further, the critic will have a small influence on improving the actor policy by moving it in the direction of the policy gradient. Therefore, the policy is not directly obtained from the value-function and, therefore, result in fewer oscillations in the policy function while also benefiting from convergence properties of the policy gradient methods (unlike critic-only methods, Grondman (2015)). The learning rate for the actor is normally set smaller than the critic so that enough time is given for the critic to evaluate the policy. The convergence criteria for the actor-critic architecture can be found in Konda and Tsitsiklis (2003). Grondman (2015) proposed the use of natural gradients instead of standard gradients for AC implementations to speed up learning without increasing the computational cost. Improving policy for each Q-function, by policy evaluation, is done by using the Bellman equation.

The advantage of this method, compared to the critic-only method, is the preservation of convergence properties that are inherited from policy gradient methods. In policy gradient methods, the estimator usually use the cumulative rewards $R_t = \sum_i \gamma^i r_{y+i}$ or approximation. The accumulated reward would result in higher variance but lower bias, while approximation can cause more bias and less variance. In approaches such as ACER (section 2.4), $R(t)$ is integrated with value function approximation in an attempt to overcome the limitations of both. For AC approaches low-variance and faster learning is expected, although a larger bias is created at the start of the learning (Grondman, 2015).

Actor-Critic with Experience Replay (ACER)

Experience Replay methods (Adam *et al.*, 2012) are introduced for efficient use of the available samples by storing the measured data such as reward, next and current states, and the actions are taken. A similar approach called prioritized sweeping for efficient use of the measured data was proposed by Moore and Atkeson (1993). As the environment in which an agent is trained to become closer to reality, the agent capabilities would improve; however, with higher simulation costs. An approach to reduce the cost of simulation is to have more efficient data available for training and hence improve sampling. This improves done on Deep Q-Networks (DQN) methods that are one of the most sample efficient methods (Wang *et al.*, 2016). This off-policy policy gradient method applies to both continuous and discrete action spaces, as an improvement in sample efficiency to existing comparable methods such as asynchronous advantage actor-critic (A3C) (Mnih *et al.*, 2016). The difference between ACER and A3C can be seen as if the former is an off-policy and the latter the on-policy approach. Apart, ACER includes engineering innovations such as truncated importance sampling with bias correction, stochastic dueling network architectures, and efficient Trust Region Policy Optimization (TRPO) (Schulman *et al.* (2015), Wang *et al.* (2016)). The challenge with off-policy methods is the stability and variance control, which is improved with ACER. In ACER, the integration of cumulative rewards R_t and value function approximation which obtained from A3C is combined by the gradient approximation to computed the update error:

$$\hat{g}^{a3c} = \sum_{t\geq 0}((\sum_{i=0}^{k-1}\gamma^i r(t+i)) + \gamma^k V_{\theta_v}^{\pi}(x(t+k)) - V_{\theta_v}^{\pi}(x(t)))\nabla_\theta log\pi_\theta(a(t)|x(t)) \tag{2.11}$$

The method use retrace approximation for the recursive approximation of the value function, which results in less bias in policy gradient and faster learning for critic and, therefore, less bias in total (Wang *et al.*, 2016). Also, the trust region policy optimization (TRPO) method is used for ensuring stability for e.g., occasional substantial updates. Therefore, ACER is a modification of A3C, in which Q-functions are used instead of value-functions, and TRPO methods are implemented. It is shown that ACER with four replays is comparable in its performance to prioritized DQN and A3C. TRPO is One of the most critical baselines in model-free continuous control problems (also applicable to discrete domains). In general, using experience replay, which is possible by storing the past experiences of the controller during training and then randomly choose between the batch of the experiences, would solve the problem of TD correlations and variances. Besides, using a set of target networks for computing TD error increases stability in the learning.

Model-learning actor-critic methods

In some approaches, where the learning of the model f is required, the model-learning AC methods can be used as an indirect learning algorithm. The discrete DYNA algorithm originally introduced this idea and based on the certainty equivalence principle (Sutton, 1990). The absence of a model that comes from the model-free principle of RL algorithms implies a long time learning as an optimal policy can be found if enough knowledge is gathered about the system. This information that comes from interaction with the environment, therefore, should be collected as efficient as possible to reach the optimal policy in a short time. For most of the RL approaches, the source of information is the transition sample at each time step. Grondman (2015) tried to find a relation between the transition samples collected during learning and derive an approximation of the model. This relation is then used to construct an algorithm that predicts the system's behavior, using interpolation and/or extrapolation. It is shown that this approach increases the speed of learning.

In the same study, the reward function is used explicitly in the transition sample in an attempt to increase learning speed by making use of knowledge about the desired closed-loop behavior. It is noted by the author that although a process model is learned by the RL controller during training, the model-learning approach can still be assumed model-free since no knowledge about the system is required in the prior to the learning. In the same research, it is shown that using local linear regression for the approximation of the model of the system in a model-learning approach can reduce the learning time by 20-50 % when compared to entirely model-free AC methods, which also use approximation. Different categories of these model-learning methods are described in section 2.5. In some literature, a demonstration is used for learning a model of the system, while in others, system identification methods are used.

2.3.4. Exploration vs exploitation

The principal challenge in RL methods is the exploration-exploitation: the agent must experience as much as possible within the environment to learn its behavior (Russell and Norvig, 2016). Exploration denoted to experiencing the uncharted territories and exploitation corresponds to the use of the current knowledge. Particularly, in an online performance, finding a balance between the two become critical. An RL agent should always make a trade-off between exploitation, maximization of the instant reward, and exploration, maximization of the long-term prices, a study for multi-armed bandit problems and finite MDPs. Not exploring enough can result in the agent adhering to a certain path without seeing other opportunities. Pure exploration can also enlarge the computational time and jeopardize practicability. Therefore, efficient exploration is an important factor in the convergence time of the learning algorithm. Delayed Q-learning is one of the approaches for a model-free online learning exploration method (Strehl *et al.*, 2006). It should be noted that in an RL method with enough exploration, the performance will not increase monotonically and might even degrade temporarily due to exploration. Exploration in continuous RL can sometimes be introduced to the system with a noise term.

2.3.5. Reward function

In classical RL problems, the reward/cost function is part of the controller's system and unknown. One of the developments for modern RL is the direct accessibility of the controller to the reward/cost function. Therefore, the controller now gets information about the rewards/costs per time step (sample) basis. This incremental reward function gives a better presentation of the influence of the policy. The definition of the reward function as one of the certain elements of the RL problem is presented in section 2.2. Since a good design for reward function is an essential and nontrivial step in implementing RL-based control, this section is dedicated to this topic. The reward function for many problems is usually designed by the engineer. Therefore the explicit knowledge of the reward is directly available to the controller. In early literature, it is recommended to make the reward function as simple as possible and even in some research which deals with reaching a goal in the fastest way possible like the mountain-car problem the agent is given a positive credit only if the final goal is reached (Sutton and Barto, 1998).

For an online platform, however, a simple reward function might slow RL learning. Therefore, using a reward function that would include more information might result in more successful learning. Also, for a higher level of control in automated systems, the controller behavior, in addition to the goal, should be considered. This includes requirements on overshoots or the rate of the convergence to a stabilized condition, etc. Such requirements can also be translated into the reward function, although it can be very challenging. Still reward function is the best platform to encode domain knowledge to the RL controller (Dorigo and Colombetti (1994), Randløv and Alstrøm (1998), Ng *et al.* (1999)). The *Return* is then calculated as the discounted sum of the rewards or the average of the reward received by the agent (Grondman, 2015). A comparison of how the shape of the reward function can influence the performance of the RL controller in a discrete inverted pendulum problem is done by Doya (2000) that some of the results are presented by Figure 2.3. It can be seen that for the swing-up pendulum problem, the reward function with cosine shape needs the least number of trials to achieve the target of stabilizing the pendulum (this will be used later for generating the preliminary results).

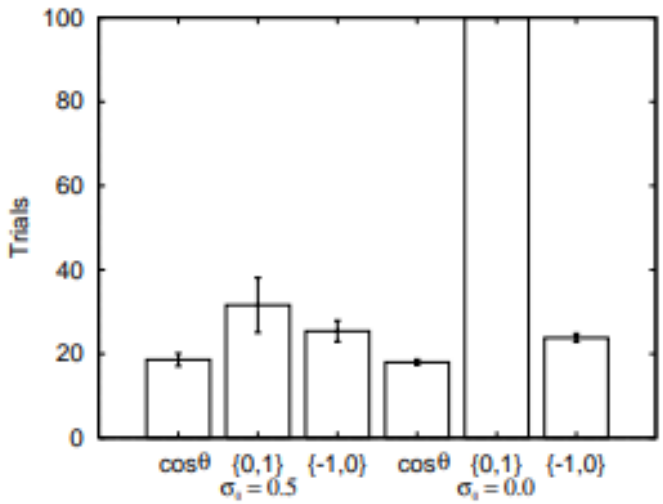


Figure (2.3) Effect of policy parameters and perturbation noises on reward function in number of trials required to stabilize the pendulum (Doya, 2000)

2.3.6. Online vs offline algorithms

In an offline algorithm, the agent learns policies from the batch data and only when all the data become available to the agent. However, in online mode, the policy update occurs after some specific amount of data becomes available to the agent, i.e., after every K_θ time steps. If this time step becomes small enough, so that the update happens after every time step, then the algorithm gets close to the real-time. In practice, a fully real-time algorithm cannot be reached for RL, however an online implementation can be used (Bertsekas *et al.* (1995), Palunko *et al.* (2013)).

In some online algorithms, to make sure that adaptation to the change in the condition would be achieved, a forgetting factor can be used for online algorithms. This forget factor put priorities on the recently obtained data and hence increase their influence on the policy update. One of the ways to cope with adaptation is the use of the forgetting factor. Since the agent uses the info gathered in offline and online phases, in the occurrence of a sudden change, the algorithm might use some experiences in the past that are no longer valid. With the use of the forgetting factor, more weights will be given to recently acquired data. For an RL agent, this forget factor illustrate itself in the existence of the discount factor.

Offline algorithms, in general, exploit the data in a more efficient manner comparing to online approaches. On the other hand, data efficiency in online algorithms can be improved with eligibility traces Sutton and Barto. A combination of two, which is achieved by interspersing offline updates with online operations, an approach called pseudo-online, can lead to an algorithm that also satisfies data efficiency.

2.4. Reinforcement Learning Methods Classifications

RL methods can be classified on different terms. As proposed in the latest edition of Sutton and Barto (2018), the modern RL algorithms can be categorized mainly into tabular or discrete RL problems and approximate or continuous RL problems. Although the basis of the two approaches is the same, however, they differ in the nature of the problem they are dealing with (section 2.2). Tabular methods are suitable for problems with small state and action space size. To improve the continuity in the transfer of learning experiences between tasks or in higher freedom dimensions, continuous RL methods can be exploited. In the following, these two categories of RL solutions are illustrated with the aim to show the requirement for approximate methods.

2.4.1. Classical reinforcement learning problem

In this study by classical RL, it is meant methods that work with discrete states and spaces. These methods can result in the problem of dimensionality and introduce hidden states while also eliminating the Markov property (Glaubius *et al.*, 2005). Several reasons result in the need for continuous implementations as illustrated by Doya (2000). First, coarse discretization can result in poor performance control outputs that are not smooth. Discretization can result in a large number of states which increase the number of iteration steps. As a result, large memory storage is needed, and a larger number of learning trials are required to achieve the desired performance. Finally, prior knowledge is required to find suitable and enough accuracy for partitioning the variables.

One of the widely used methods for problems with discrete states and actions is Q-learning. Is such problems, the state-action spaces, and the reward function are combined in a tabular cell of Q-function magnitude. A challenge was presented in modifying of the *classical* Q-learning to problems that deal with continuous state and spaces (Gaskett *et al.*, 1999). The common approach of splitting the state domain to discrete regions arose the problem of aliasing and lost of information. Then again, fine discretization leads in curse of dimensionality (Naruta, 2017). Bellman (2013) defines the curse of dimensionality as the memory cost of the RL problem, which is proportional to n^l where n is the resolution of the state and action spaces and l is the dimensionality of the state. To optimally control complex systems with a large number of states and with no prior knowledge of the system, approximate RL will be used.

2.4.2. Approximate reinforcement learning problem

As the size of the state-space increases, and for the continuous action spaces (which is mostly the case in aerospace applications), the generalization of the state-space can help with the convergence of the optimal policy in a reasonable time scale (Ravishankar and Vijayakumar, 2017). The computational intensity and the tuning of the function approximators parameters are the most important challenges in continuous RL problems. In addition, the theoretical guarantees for RL systems when using approximation can be weaker compared to situations when the approximation is avoided. In some studies the Q-learning and TD methods show divergence when approximation is integrated (Baird (1995), Bertsekas *et al.* (1995), Perkins and Barto (2002)). However, approximate RL also comes with lots of advantages, e.g., a control performance achieved by approximate RL will be smooth while an efficient control policy can be achieved. In addition, the granularity of the state and action spaces can be left to the approximation method and the numerical integrations while the generalization can happen to states that have not been visited even once (Naruta, 2017). In optimal control literature, continuous RL is mostly introduced by the term of approximate dynamic programming. For studying the history of development of continuous framework for reinforcement learning the reader is refer to Peterson (1993), Munos (1997), Munos and Bourgine (1998) and Dayan and Singh (1996).

Approximate dynamic programming

To overcome the curse-of-dimensionality in RL control problems caused by backward-looking in solving the optimal control problems, the Approximate Dynamic Programming (ADP) was introduced. To address the limitations, algorithms such as Neuro-dynamic Programming (NDP) which employ Neural Networks (NN) to give approximate solutions to Hamilton-Jacobi-Bellman (HJB) equations forward-in-time, were introduced (Liu *et al.*, 2015). Khan *et al.* (2012) see the difference of ADP to the traditional DP (section 2.3) in the latter being an offline approach which solves the optimality problem backward in time, while ADP is the online approach with a forward look. Conventional dynamic programming methods need a model of the system. Unlike classical techniques for automatic control, the approximate methods make few assumptions on the system and therefore are applicable to stochastic and nonlinear systems.

The generalization for an ADP problem happens with the use of Function Approximation (FA) methods. The choice of the FA is not trivial. When searching for a suitable FA, the complexity should be accounted, since it directly influences the memory and computational costs (Busoniu *et al.*, 2010). In addition, increasing the complexity of the approximator also suggests an increase in the amount of

data required for designing the approximator. However, for especially online applications, which is possible with model-free algorithms, usually, the knowledge of the system is not available, hence the focus in these applications would be on low-complexity approximators. The approximation can happen in parametric or non-parametric forms. A parametric estimate can be neural networks and their use of radial basis functions. In non-parametric case, kernel-based approximations are typically used. One of the typical examples is Gaussian kernel-based methods. The function approximation can be used for estimation of value function and policy since they are not explicitly time-dependent, as can be seen by an example representation of policy and value functions by Equation 2.12 and Equation 2.13. Parametric FAs are described later in this chapter by subsection 2.4.3.

$$u(t) = \pi(x(t)) \tag{2.12}$$

$$V^{\pi}(x(t^{'})) = \int_{t}^{\infty} e^{-\frac{t^{'}-t}{\tau}} r(x,u,t^{'})\mathrm{d}t^{'}. \tag{2.13}$$

In general, three approaches can be seen as approximation in RL: value-function approximation (value-iteration), policy function approximation (policy iteration) or policy gradient/actor-critic (PG/AC) approach. Sutton *et al.* (2000 propose two limitations with the value-iteration approach, it cannot be applied for finding the stochastic policies and its inability of convergences assurance (probably due to the discontinuous changes in value-function generalization). He showed, however, that the policy iteration with differentiable function approximators in a model-based platform can converge to a locally optimal policy. The estimate policy can then be used to approximate a proper value-function. In real-world problems, normally, the state and action are continuous, and hence the focus of approximation as a step towards practicability would be the approximation of the Q-function, which includes the state and actions. Therefore, policy approximation is not a common practice, since it can be directly computed in demand by a greedy approach. However, the algorithm must make sure to find the global optimum solution for the approximate Q-function. In approximating the Q-function, typically, the actions are discretized while the state space is approximated with basic functions. A schematic of the methods that can be used for approximation is depicted in Figure 2.4.

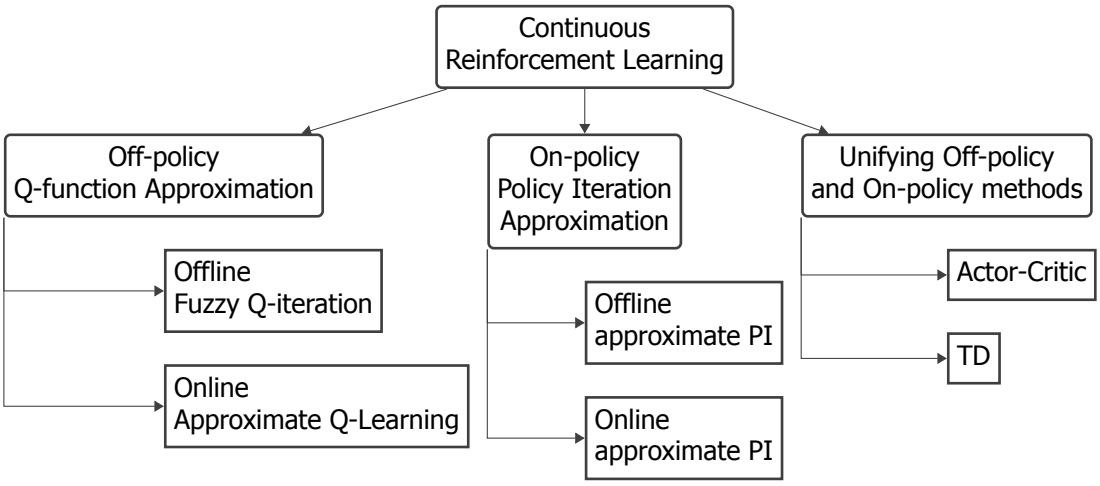


Figure (2.4) RL classification based on approximation approaches

Initially, linear model-free ADP approaches use a Linear-Quadratic Regulator (LQR) to approximate the kernel matrix based on the Temporary Difference (TD) error. However, they have the assumption for the system to be Linear Time-Invariant (LTI). They are also unable to tackle with partially observable systems, and since these methods are offline, they cannot handle uncertainties in the environment

(not adaptable). Therefore, feedback control approaches to increase ADP domains to adaptive control were proposed. Also, the early versions of ADP made adjustments to the value function after each iteration. However, these adjustments might not be significant enough. One approach to bound these adjustments is the use of heuristic ranking of the possible adjustments by e.g., prioritized sweeping in which the agent carries on with adjusting those states that their possible successors have just undergone a significant adjustment in their utility estimates. Using this method, the ADP approaches can learn almost as fast as the DP algorithms (in terms of training trials) while being more efficient in computation (Russell and Norvig, 2016). Also, ADP methods are more efficient while moving from the early stages of the training to the convergence by decreasing the minimum adjustment size as the environment model becomes more accurate. ADP methods in an actor-critic structure combined with TD methods and with heuristic search are categorized inspired by Werbos (1992) in intelligent control handbook of Whitford (1987) to four main groups (Khan *et al.*, 2012). Two new methods were then suggested by Prokhorov and Wunsch (1997). Therefore, the main classification is as:

- Heuristic Dynamic Programming (HDP), in which the one entity called critic directly estimate the value function.
- Dual Heuristic Dynamic Programming (DHP), in which the derivatives of the value function are estimated (Khan *et al.*, 2012).
- Action Dependent Heuristic Dynamic Programming (ADHDP), modified version of HDP with direct connection of the policy entity so the actor to the critic network (Q-learning extension to continuous state and space) which avoid knowing the system dynamics and can be seen as a complete model-free learning approach.
- Action Dependent Dual Heuristic Dynamic Programming (ADDHP), modified version of DHP with direct connection of actor network to critic network.
- Global Dual Heuristic Dynamic Programming (GDHP), in which both the estimation of the value function and its derivative is used.
- Action Dependent Global Dual Heuristic Programming (ADGDHP) with the connection between the actor output and critic input ([24], 2006)n.

where *heuristics* denotes to the knowledge based on Trial-and-error, evaluations, and experimentation. The block diagram and update rules for these methods along with detailed description are given in section 2.5.

Another ways of classifying RL algorithms exist. Busoniu *et al.* (2010) categorized RL algorithms based on model knowledge as follow:

- Model-based: in which the model of the system f and the reward function definition r are known.
- Model-free: the f and r are initially unknown for the agent
- Model-learning: estimate the f and r, using the system data and responses

Another way to classify RL methods is based on their interaction with the data. Therefore to:

- Online: Learning occurs while interacting with the environment.
- Offline: Learning occurs only after all the data is collected as a batch.

Where these categories were described more elaborately in subsection 2.3.6.

Comparison between discrete and continuous RL

One of the first studies which deal with continuous state and action spaces is done by Doya (2000). In this research, the author compared discrete and continuous RL problem of swinging up a pendulum by the learning time and the number of trials the RL agent takes to do a successful swing-up. It is shown in the study that the discrete agent takes more trials before a successful one when the simulation time steps are the same for both approaches. The spatial generalization of approximation

results in the faster achievement of the RL goal, while the discrete agent will need more resolution to attain a close performance to the continuous method. In the same study it is shown that the actor-critic approach in which the policy is improved stochastically (the update of the parameters is done by stochastic real-valued (SRV) algorithm) has the poorest performance, while the value-gradient based policy approach has the fastest learning compared to the advantage updating approach where the model is simultaneously learned. In the same study, the performance of the agent has been analyzed for different shapes of the reward function, the action definition (action cost), and the exploration parameter. It is shown that the learning rate is slower for high action costs and less consistent with no action cost (bang-bang control). Next, it is shown that although the continuous reward function (in this case $cos(\theta)$) shows better learning results compared to binary reward definition, however, the negative binary reward functions perform much better than the positive function and close to the continuous function. Using the same initialization point for the trials and excluding the exploration coefficient (σ) result in worse learning performance for the negative binary reward function, while for the two other cases, it almost remains unchanged.

2.4.3. Function approximation methods

The most crucial topic in approximate RL is the choice of a suitable Function Approximation (FA). Using function approximation in learning can be seen as a type of supervised learning when the target of the network is known and available. Most of these methods include minimization of an error term over the distribution of input vectors by adjusting the parameter vector for each data point in the direction of reducing the error gradient, e.g., in a gradient-descent approach. van Kampen *et al.* (2006) describes the properties of a good FA as the one which is less complicated and more cost-efficient than the tabular data, and is differentiable in addition to some reasonable level of accuracy. Most of the FA approaches satisfy the first two requirements. However, for an approximate RL, the last condition can be more challenging since the target function is unknown before learning. This leads to the need for an FA, which is general enough, i.e., artificial neural networks (ANN), and can approximate a wide variety of functions with limited parameters. The FAs can be divided into parametric and non-parametric and from simple approaches such as look-up tables interpolation and binary coarse coding to more sophisticated methods such as spline polynomials and ANN. The parametric category can then be categorized as the linear like RBFs and nonlinear such as neural nets (Busoniu *et al.*, 2010). Non-parametric approximators, like locally linear regression methods, refer to locally derived parameters based on the data. The training methods for FAs can be done with the following approaches: linear regression, gradient-descent, back-propagation, and Levenberg-Marquart. Gradient-descent methods are the most popular and promising in an actor-critic RL structure. A general gradient-descent method can be formulated as:

$$\bar{\theta}(t+1) = \bar{\theta}(t) + \alpha[V^{\pi}(s,t) - V_t(s,t)]\nabla_{\bar{\theta}_t} V_t(s,t) \tag{2.14}$$

In which α is the learning rate parameter and $V^{\pi}(s,t)$ is the unbiased estimate of the target function (or the prediction of the function (Naruta, 2017). In a gradient-descent approach, the function approximator parameters $\bar{\theta}_t$ are guaranteed to converge to some local optimums. The approximation function V_t can be a linear function or a quadratic function. Neural Networks and splines are other function approximators that can be used for modeling more complex and nonlinear systems. Some popular FAs are presented in the following.

Artificial neural networks

Neural Networks (NNs) are used for modeling and controlling complicated systems, and they can achieve sufficient accuracy without knowing the inner compositions of the system. They are divided into two categories of convolutional NNs and recurrent networks. Neural networks have shown to be able to approximate any functions by using multilayer networks in theory by Kolmogorov and so it is called the universal approximation theorem. The building units of a neural network typically consist of 3 main elements of inputs, hidden layers, and outputs. The fundamental component of all NN frameworks is the artificial neurons, which include the weight summation of the input data to the network and can also add a bias term (hidden layer). The results then pass through an activation function that comes as various forms, depending on the application of the neural nets. The sum of the weighted

hidden layers results in the output of the network.

Neural nets are closely linked to different approaches of AI learning hysterically, in particular, reinforcement learning. This makes ANN a natural mechanism to use in association with reinforcement learning. Also, the application of the NN in feedback control has been also accelerated recently (Modares *et al.*, 2013). The integration of NN with dynamic programming is known as Neuro-Dynamic Programming (NDP) (Tsitsiklis and Van Roy, 1996). All this, in addition to the ability of NN to approximate various functions with a limited number of parameters, makes them currently the most often used FA for implementation of the AC architecture. In theory, a feed-forward NN with even one hidden layer and sigmoid activation functions can approximate any continuous nonlinear arbitrary function.

A typical neural network adjustments include the choice of the activation functions, the layout of the neural network, and the training method. One of the difficulties with neural networks based system identification is tuning the parameters of the model to result in a suitable approximation. The typical activation functions are threshold functions, radial basis functions, tangent-sigmoidal, and rectifiers (relu) functions. The main layouts for neural nets are Feed-Forward (FF), and Radial Basis Functions (RBF), where the former is a global approximation, and the latter is considered a local approximator. What makes NN different from other approximators, such as the fuzzy approach, is their complexity due to the need for training and optimization. However, it is shown that NN, with even a small number of neurons, can achieve more accurate results compared to fuzzy systems (Xie *et al.*, 2012). One problem that may arise for using neural networks for problems with a large number of steps is the run-time which increases exponentially with the increase in the number of radial basis functions.

The universal approximation property of the NN indicates that with using a two-layer NN with suitable weights on a boundary set of $x \in \Omega$, any function $f(x(t))$ can be approximated as:

$$f(x(t)) = W_{jk}^T \phi(W_{ij}^T x(t)) + \epsilon(x(t)) \tag{2.15}$$

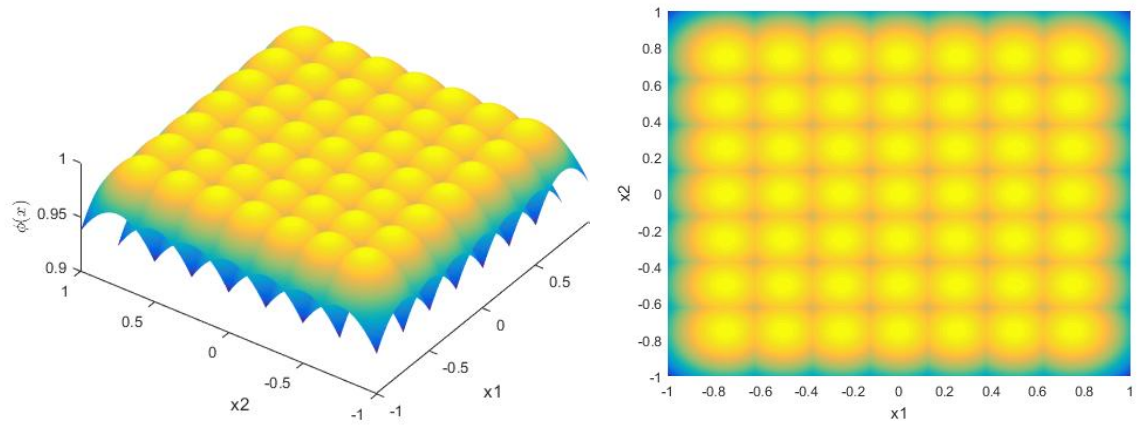


Figure (2.5) A schematic of the RBFs in neural network approximation

Where $\phi(x(t))$ is the activation function, W_{ij} is the weight matrix for the first layer (input) weights and W_{jk} is the weight matrix for the second-layer or (output) layer if a linear output transfer function is chosen. $\epsilon(x(t))$ is the NN functional approximation error. If the number of NN neurons are chosen sufficient enough, $\epsilon(x(t))$ is bounded for all $x \in \Omega$. In some literature, e.g. by Modares *et al.* (2013) and Vamvoudakis and Lewis (2010), it is shown that with fixing the input layers weights to reduce the complexity of adjusting the nonlinear in parameters input weights and by only tuning the output layers weights a good approximation can be achieved if boundaries are defined for the activation function and

its gradients in addition to the FA error and its gradients. The distribution of neural nets basis functions for approximation of a hypothetical function of $f(x1, x2)$ with states defined by $x \in x1, x2$ are sketched in Figure 2.5.

Cerebral Model Articulation Controller (CMAC)

One of the interesting approximation methods used by Naruta (2017) for controlling a quadrotor in the continuous state and action spaces with online policy iteration methods is the Cerebral Model Articulation Controller (CMAC). The advantage of this method over NN is that only the local weights, in the proximity of the state, are activated and updated in forwarding and backward passes, respectively. This approach is known as state coding. State coding can also be applied for NN where neurons center are situated in a regular grid. By defining activation regions for the NN neurons, the efficiency of the NN can be improved largely. However, this approach is more beneficial for multi-dimensional input states and networks that require a high number of hidden layer neurons. The use of activation functions is also eliminated in the CMAC approach, and the output update is done simply by summing up the weights. However, the primary benefit of RBF NN is the use of activation functions that lead to an optimization process that results in higher computational performance that would rule over the CMAC approach. As a general comparison, NN shows higher approximation power, while CMAC can be computationally simpler.

Fuzzy systems

Fuzzy systems have been used for approximation of the nonlinear systems. The main advantage of these methods compared to neural networks is that they do not need training and hence, are simpler to implement. Fuzzy systems include three parts: fuzzifier, fuzzy rules, and defuzzifiers (Xie *et al.*, 2012). There are two classifications for fuzzy systems: the Mamdani fuzzy system (Mamdani, 1974) and the Takagi, Sugeno, and Kang (TSK) fuzzy system (Sugeno, 1985). The main differences between these two approaches are in the normalization. The first part converts the inputs to fuzzy variables with values between 0 and 1 using membership functions such as Triangular and trapezoidal. In the next step, the fuzzy rules are performed in the form of min and max operators on the fuzzy variables in the Mamdani approach and in the way of min in the TSK approach. In the last step of defuzzifiers, the results of the operators and converted to analog outputs. neuro-fuzzy systems have been designed to overcome the deficiencies of the two separate approaches. However, these methods have more computational complexities while almost having the same performance.

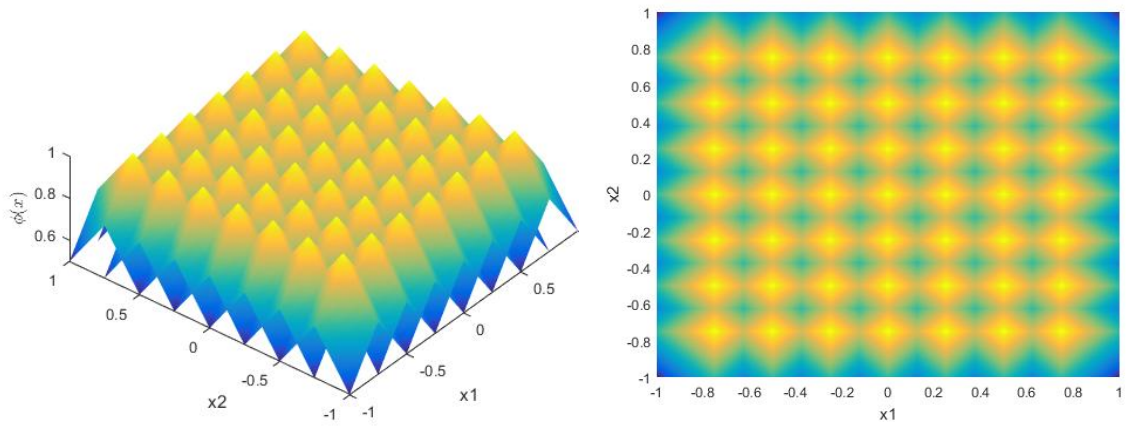


Figure (2.6) A schematic of the fuzzy basis functions

Multivariate splines

Another statistical method of approximation which divides the training data to separate piece-wise linear segments (splines) with slopes or differing gradients. Spline modeling performance depends on

proper spline space definition. The spline models are suitable for systems with nonlinear behaviors but with a trade-off on the complexity and can be stuck in the curse of dimensionality. In recent research by Eerland *et al.* (2016), the combination of ANN and splines with dynamic programming and RL are studied and compared. The main difference of ANN with spline models is that NN is a nonlinear-in-parameters approach which spline is linear-in-parameters. Recursive Least Squares with TD methods are used as the updating rule of the FA parameters in the online learning framework. It is shown in this study that while spline approximation can result in lower approximation power when compared to NN approximation, it is also limited to problems with a maximum of two number of input states.

2.4.4. Solution to approximate reinforcement learning

In the following, it is illustrated how policy evaluation is formulated in continuous RL framework, inspired by Doya (2000) and Grondman (2015). If the current estimate of the value function is given as:

$$V^{\mu}(x(t)) \simeq V(x(t);\theta), \tag{2.16}$$

where θ is the parameters of the function approximator. The update of the value function estimation, considering TD learning, can be done locally by differentiating Equation 2.16 with respect to t using the chain rule of $\dot{V}(t) = \frac{\partial V}{\partial x}\dot{x}(t)$ as:

$$\dot{V}^{(}x(t)) = \frac{1}{\tau}V^{\mu}(x(t)) - r(t), \tag{2.17}$$

Equation 2.17 is called the consistency condition. To check whether this condition is satisfied, the TD error is defined:

$$\delta(t) = r(t) - \frac{1}{\tau}V(t) + \dot{V}(t) \tag{2.18}$$

If consistency is not satisfied, the approximation should be modified in the direction to decrease Equation 2.18. Therefore, the objective function for the optimization problem can be given by:

$$E(t) = \frac{1}{2}|\delta(t)|^2 \tag{2.19}$$

Therefore the gradient descent algorithm for updating the function approximator parameters are given by:

$$\dot{\omega_i} = -\alpha\frac{\partial E}{\partial w_i} = \alpha(t)[\frac{1}{\tau}\partial\frac{V(x;w)}{w_i} - \frac{\partial}{\partial w_i}(\frac{\partial V(x;w)}{\partial x}\dot{x}(t)] \tag{2.20}$$

where α is the learning rate and determines how much of the past learning is retained. The TD error can be updated by TD(0) and TD(λ) methods. The former, also known as residual gradient, is based on backward Euler differentiation which results in:

$$\delta_t = r(t) + \gamma V(t) - V(t-1) \tag{2.21}$$

if the discounted factor value γ is close to $1 - \frac{\Delta t}{\tau}$ and $V(t)$ equals to $\frac{1}{\Delta t}V(t)$. Or if V is the value function at state x defined by $V_\theta(x) = \theta^T\phi(x)$ the update rule can be set as:

$$\delta_k = r(t+1) + \gamma V_{\theta_t}(x(t+1)) - V_{\theta(t)}(x(t)) \tag{2.22}$$

Another method for updating the parameters of the function approximator is by TD(λ) method, also known as exponential eligibility traces. Then, the learning algorithm can be made as:

$$\dot{w}_i = \eta\delta(t)e_i(t)\dot{e}_i(t) = -\frac{1}{k}e_i(t) + \frac{V(x(t);w)}{w_i} \tag{2.23}$$

where $0 < k \leq \tau$ is the eligibility trace constant. Equation 2.23 can be written in discrete form as:

$$e_i(t+\Delta t) = \lambda\gamma e_i(t) + \frac{V(t)}{w_i} \tag{2.24}$$

In which $\lambda = \frac{1-\Delta t/k}{1-\Delta t/\tau}$ and e is the eligibility trace vector. The Equation 2.24 with TD error and gradient descent approach can be rewritten as:

$$e_i(t+\Delta t) = \lambda\gamma e_i(t) + \nabla_\theta V_{\theta_t}(x(t)) \tag{2.25}$$

Based on Equation 2.20 the critic parameters (weights) are updated by:

$$\theta(t+1) = \theta(t) + \alpha_c\delta_k e(t) \tag{2.26}$$

If continuous action is assumed which will be updated by function approximation methods such as neural networks as $\pi_v(x) = v^T\phi(x)$, the policy improvement will be defined as:

$$v(t+1) = v(t) + \alpha_a\nabla_v J(t) \tag{2.27}$$

Normally the gradient will be defined as $\nabla_v J_k = J - J^*$ (more information on section 2.5. In critic-only methods, the Q-function is learned by TD error minimization, defined as:

$$\delta(t+1) = r(t+1) + \gamma Q_{\theta(t)}(x(t+1), u(t+1)) - Q_{\theta_t}(x(t), u(t)), \tag{2.28}$$

The parameters of the Q-function are updated with gradient descent methods by:

$$\theta(t+1) = \theta(t) + \alpha_c\delta(t+1)\frac{\partial Q_{\theta_t}(x(t),u(t))}{\partial\theta} \tag{2.29}$$

In such a method, the policy will be updated not by an FA but with an optimization procedure over the value function:

$$\pi(x) = argmax_u Q(x,u) \tag{2.30}$$

An $\epsilon - greedy$ action selection can be used for policy to include exploration. If eligibility traces are used, the update will be:

$$e(t) = \lambda\gamma z(t-1) + \frac{\partial Q_\theta(x(t), u(t))}{\partial\theta}, \tag{2.31}$$

$$\theta(t+1) = \theta(t) + \alpha(t)\delta(t+1)e(t) \tag{2.32}$$

What mentioned above is the general framework for solving most of the continuous RL approaches.

2.4.5. Deep reinforcement learning

Deep reinforcement learning has recently attracted AI and ML scientists. With the introduction of Deep Q-Networks (DQN) by Mnih *et al.* (2015), the ability of deep learning methods in solving high-dimensional problems become evident as it is a challenge for RL-based control community to handle continuous action and state spaces. To solve this problem, Google DeepMind has introduced the Deep Deterministic Policy Gradient (DDPG), which is an AC/PG algorithm (Lillicrap *et al.*, 2015). Deep learning has been widely used for generalization and pattern recognition in supervised learning in recent years.

Deep Q-Networks (DQN)

A new direct learning algorithm which is independent of learning an explicit model of the system is known as Deep Q-networks (DQN). In such a method, the optimal state and action value function or Q-function are approximated using deep neural-networks (i.e., with more than one hidden layer). The weight update rule for the network is made by minimizing the expected TD error from Bellman's equation using stochastic gradient descent and back-propagation. Also, a buffer is used to store the MDP tuple. Target networks are used to store the parameters for the target Q-values in addition to parameters of the current Q-function estimate. The policy evaluation and improvement in a DQN framework are combined, which might result in over-estimation of the Q-values. Double DQN, which makes a separation between PE and PI, is proposed to overcome this problem.

Deep Deterministic Policy Gradients (DDPG)

Deep Deterministic Policy Gradient (DDPG) is an off-policy and model-free method that use a combination of DQN and deterministic policy gradients (Silver *et al.*, 2014). To be more specific, DDPG is a policy gradient algorithm in which the stochastic behavioral policy is used for exploration, while a deterministic target policy will be estimated. DDPG algorithm uses the same architecture as actor-critic methods with two neural nets for approximating the actor and critic networks with the critic network output be the Q-function estimation and the actor network output the action chosen from continuous action spaces. The update rules for the critic network are the gradient of the TD error signal while the actor is updated from the deterministic policy gradient theorem. DDPG uses experience replay for decreasing the variances in the Q-function estimation caused by TD correlations in different training episodes. It also uses target networks to increase stability in learning and avoid divergence. Although deep RL methods show promising results for more straightforward problems such as swing-up pendulum with two states, it should be noted that the tuning the parameters of the NN become more complicated when compared to actor-critic methods with NN approximation with only one-layer of neurons. This becomes an important issue when the system under control is also complicated.

2.4.6. Research on different platforms available for RL implementations

Python is the first programming language that can be used for RL applications due to the existence of many libraries, and the well-known Tensorflow which became available by Google Brain. However, at this point in the thesis, choices have to be made for the proper platform based on the available models. ICE model is only available in MATLAB SIMULINK, plus the recently developed spline model. These implementations are in MATLAB; hence, this is an important consideration to be made when choosing the RL platform. Some researches were done to find the proper API between MATLAB and Python. So far, few and uncompleted implementations are found to be available, which makes the integration of Python with MATLAB hard at this moment. The next choice would be to use C/C++ for

faster computations. The reason is that using the integrated language such as C/C++ increases the run time, while it makes other advantages possible (this conclusion is made in the researcher study so far and might not be universal).

2.5. Adaptive Critic Designs

This section separately focus on a group of RL methods known as Adaptive Critic Designs (ACD). ACD methods are also known as actor-critic or action-critic designs. Although some of the techniques discussed here can be interpreted close to the methods already mentioned, however, due to their importance in ADP literature and previous studies at Control and Simulation department of TU Delft, they will be mentioned in a separate section.

A group of methods referred to as ACD, are actor-critic methods in which, classification is done if the critic element is approximating the expected total future reward or the value function and/or the derivative of the value function. The state information obtained by the critic will be used for determining the value of the state and change the internal parameters of the critic so that this knowledge will be stored in a compact way within the critic parameters. The actor, on the other hand, represents the policy π of the controller by providing actions based on the current information of the state. By making the approximated value function $J(t)$ to a goal value $J^*(t)$ using the actor training error. ACD methods with approximation can be seen as an extension of Generalized Policy Iteration (GPI) (Sutton and Barto, 2018) to continuous state and action spaces. In a GPI framework, the PE step is not performed completely, and the current estimation of the value function would be updated towards the value of the given policy (Modares *et al.*, 2013). However, in an actor-critic structure, both PE and PI steps would be done completely and simultaneously at each time step. Since the TD error is used as the update rule at each time step, it is not needed to wait for the end of each episode, which makes the AC structures suitable for online application. In addition, no greedy selection is performed, and it is the policy itself which determine the direction. One downside of such an approach which is inherited by policy gradient methods is the probability that you might trap in a local optimum if NN is used. One of the solutions can be to use softmax functions or by using TD prediction methods (e.g. using target networks). In the following the classification of ACD algorithms are presented.

2.5.1. Heuristic Dynamic Programming (HDP)

In a Heuristic Dynamic Programming (HDP) framework, the actor output is not directly fed to the critic network. However, when the critic uses only the incoming state information for value-function approximation, while the actor network is providing the control input, there should be some form of information exchange between the two networks. Since the actor needs the information updates in the critic as its output indirectly influences also the critic output. Therefore, in such cases, an internal model of the plant is approximated and will be updated along with critic and actor networks to identify and reflect the possible changes in the real plant (Si *et al.*, 2004). The optimal value-function is unknown and therefore cannot be used as the target value for the learning, although the recursive characteristics of it help the network move in the direction of the optimal value function. For HDP an indirect path from the actor output to the approximated plant dynamics and then through the critic will be used to approximate the value function, and through the same path, the actor network will be updated in direction to achieve zero value for actor error $E_a(t)$ (van Kampen *et al.*, 2006). While critic is used for approximation of the value function, the unbalance in the Bellman's equation will be used as the learning error for the actor that will need to be minimized. The actor would then lead the system to the regions within the state space that the value function approximation of the critic has high values. Therefore, the difference between the evaluated function approximation $J(t)$ and the predefined value-function goal $J^*(t)$ (e.g. designed by expert knowledge considering the chosen reward function $r(x)$) is used for defining the actor error $e_a(t)$ as:

$$e_a(t) = J(t) - J^*(t) \tag{2.33}$$

Since it is assumed that the value-function is unknown, it is best to define the reward function in such a way that all value of reward except for the goal is negative, and the maximum reward is set to zero. In such a setting, the value-function goal can be assumed to be equal to zero (van Kampen *et al.* (2006), Enns and Si (2003)). The square of the actor error is given by:

$$E_a(t) = \frac{1}{2} e_a^2(t) \tag{2.34}$$

For the critic, the error equals the TD error and is defined by:

$$e_c(t) = r(t) + \gamma J(t+1) - J(t) \tag{2.35}$$

The square of the critic error is defined as:

$$E_c(t) = \frac{1}{2} e_c^2(t) \tag{2.36}$$

By knowing the effect of the actor and critic network parameters on the training error, they can be updated in the direction of minimizing the errors using back-propagation techniques. The actor weights are then updated by:

$$\frac{\partial E_a(t+1)}{\partial w_a(t)} = \frac{\partial E_a(t+1)}{\partial e_a(t)} \frac{\partial e_a(t+1)}{\partial J(t)} \frac{\partial J(t+1)}{\partial x(t)} \frac{\partial x(t+1)}{\partial u(t)} \frac{\partial u(t)}{\partial w_a(t)} \tag{2.37}$$

The relation between critic weights and the training error is written as:

$$\frac{\partial E_c(t+1)}{\partial w_c(t)} = \frac{\partial E_c(t+1)}{\partial e_c(t)} \frac{\partial e_c(t+1)}{\partial J(t)} \frac{\partial J(t+1)}{\partial w_c(t)} \tag{2.38}$$

In Equation 2.37 each partial derivative corresponds to a part of the signal path from the objective function of the actor to its network weights. $J(t)$ derive from critic network, and the $u(t)$ came from actor network layouts. The relation between states $x(t)$ and actions $u(t)$ need to be determined. This is where the need for plant dynamics approximation becomes evident. Since the true plant dynamics is unknown to the controller, a new approximation of the plant dynamics is needed that has the control input and the previous state info as the input and can generate the approximated next state. As van Kampen *et al.* (2006) described for an actual plant dynamics defined by $x(t) = f(x(t-1), u(t-1))$ the plant approximation (e.g. by NN) will become $f'(x(t), u(t), W_p(t)) \approx f(x(t), u(t))$, where W_p are the NN weights for the plant dynamics FA. Therefore, now, the partial derivative for plant dynamics that can be used during online learning in Equation 2.37 can be defined by:

$$\partial x(t+1) = lim_{h \to 0} \frac{f'(x(t), u(t) + \Delta u) - f'(x(t), u(t))}{\partial u(t)} \tag{2.39}$$

It should be noted that the only place where the plant dynamics will be used is in back-propagation step for the actor network. Also, in such an approach, there is no direct connection between the actor output to the critic network, and therefore, the critic is estimating the value-function now only based on the state information. Therefore, the task of the critic is simplified, and the complexity is now partially relocated to the plant dynamics. Next, it will be shown that in an ADHDP structure, the plant dynamics is implicitly defined within the critic network and is directly influenced by the actor network, while in the HDP framework, the plant network is completely independent and it itself would influence the actor and critic networks. Very recently, Helmer *et al.* (2018) proposed a gradual learning method, called flexible Heuristic Dynamic Programming (FHDP) for continuous state/action space model-based

RL. The gradual behaviour is achieved by exposing the agent to a limited set of states and action variables at the beginning of the learning process. He uses sub-spaces of the state space with a limited subset of Markov Decision Process variables for learning the agent. He developed the Flexible Function Approximators (FFA) which update locally to enable handling the extension in the state space. He showed that for a 2-dimensional hover control objective of a quadrotor, the agent is able to achieve a ”better policy” while being faster than if it was exposed to the whole state space at the start. A schematic of an HDP architecture is given in Figure 2.7.

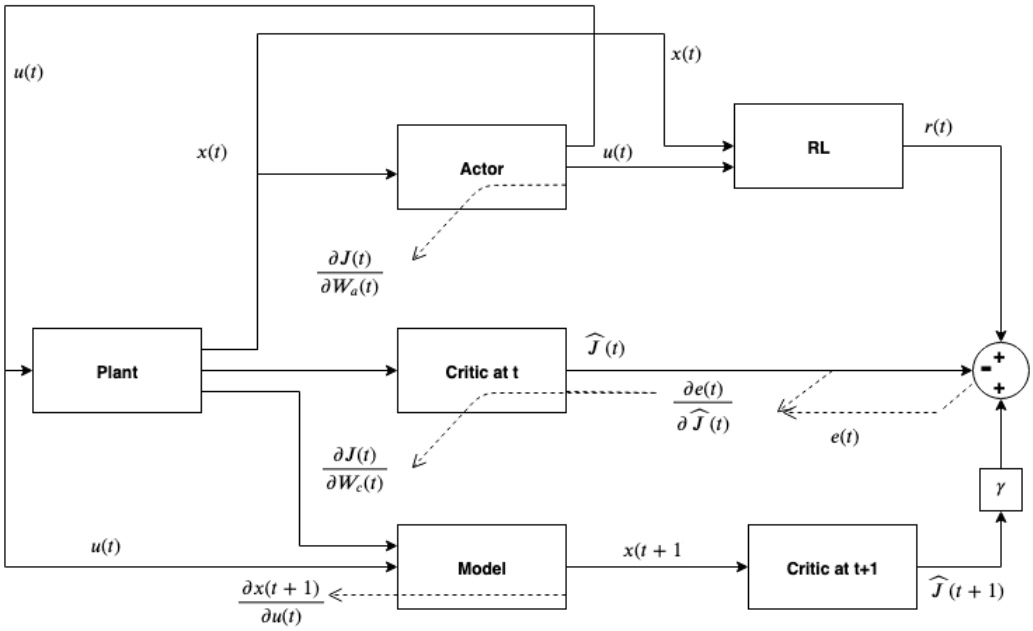


Figure (2.7) HDP scheme block diagrams and updates based on (Sutton and Barto, 2018)

2.5.2. Action Dependent Heuristic Dynamic Programming (ADHDP)

If a direct connection exists between the value function and the control action, this class of ACDs is said to be action dependent as will be discussed in the next section. In Action Dependent Heuristic Dynamic Programming (ADHDP) architecture, the actor output is directly fed to the critic. ADHDP can be assumed as one of the three main types of ACDs.

One of the arguments that discuss the advantage of a direct connection of the actor input to the critic is that the value-function is indirectly influenced by the actor output since the action provided with the actor generates the next state of the plant, and therefore result in a change in the value function. Accordingly, the plant dynamics is implicitly stored in the critic network. However, the design task of the critic is different. Although in this method, the critic eventually will be able to find a good approximation of the value-function, however, the desired mapping of state to value function will not obtain. Therefore, the HDP framework is more reasonable in order to give as the input to the critic network only the signals that have a direct influence on the approximation, hence, the state information and the desired state ([4], [4]). If the control action is defined in the reinforcement learning signal (reward function), e.g. to minimize the control effort, the ADHDP framework might become useful. The actor weights for the ADHDP are then updated by:

$$\frac{\partial E_a(t)}{\partial w_a(t)} = \frac{\partial E_a(t)}{\partial e_a(t)} \frac{\partial e_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w_a(t)} \tag{2.40}$$

Equation 2.40 presents the advantage of the ADHDP framework as the back-propagation path of the training error to the network weights is directly available through the critic network updates, and an approximation of the plant dynamics with an internal model is not required in this approach. Such a

controller is used for AH64-Apache Helicopter control by Enns and Si (2003). The schematic of the relations and updates for ADHDP architecture is shown in Figure 2.8.

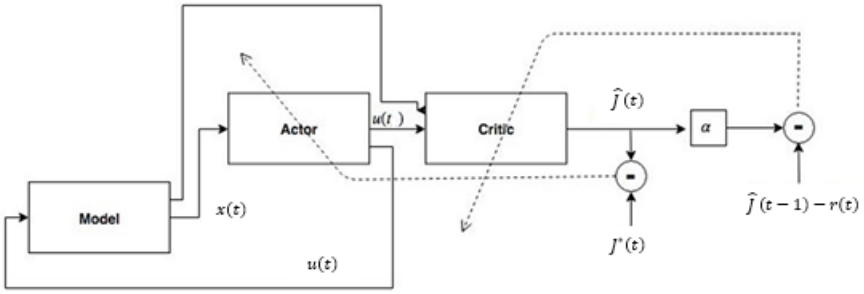


Figure (2.8) ADHDP scheme block diagrams and updates based on (Sutton and Barto, 2018)

2.5.3. Dual Heuristic Dynamic Programming (DHP)

An alternative approach to HDP is that the approximation will be used for the derivative of the value function with respect to the state, instead of estimate of the value-function itself. The expected output of the critic is defined as:

$$CriticOutput: \frac{\partial r(t)}{\partial x(t)} + \gamma \frac{\partial f(t)}{\partial x(t)} \tilde{G_{t+1}} + \frac{\partial u(t)}{\partial x(t)} \left(\frac{\partial r(t)}{\partial x(t)} + \gamma \frac{\partial f(t)}{\partial u(t)} \tilde{G}_{r+1}\right) \tag{2.41}$$

Where $\tilde{G}(t) = \frac{\partial J(t)}{\partial x(t)}$ and $r(t)$ is the reinforcement signal. The error for actor training is then defined by:

$$Actorerror: \left(\frac{\partial r(t)}{\partial u(t)} + \gamma \frac{\partial f(t)}{\partial u(t)} \tilde{G}_{t+1}\right) \tag{2.42}$$

The block diagram for this method is shown in Figure 2.9.

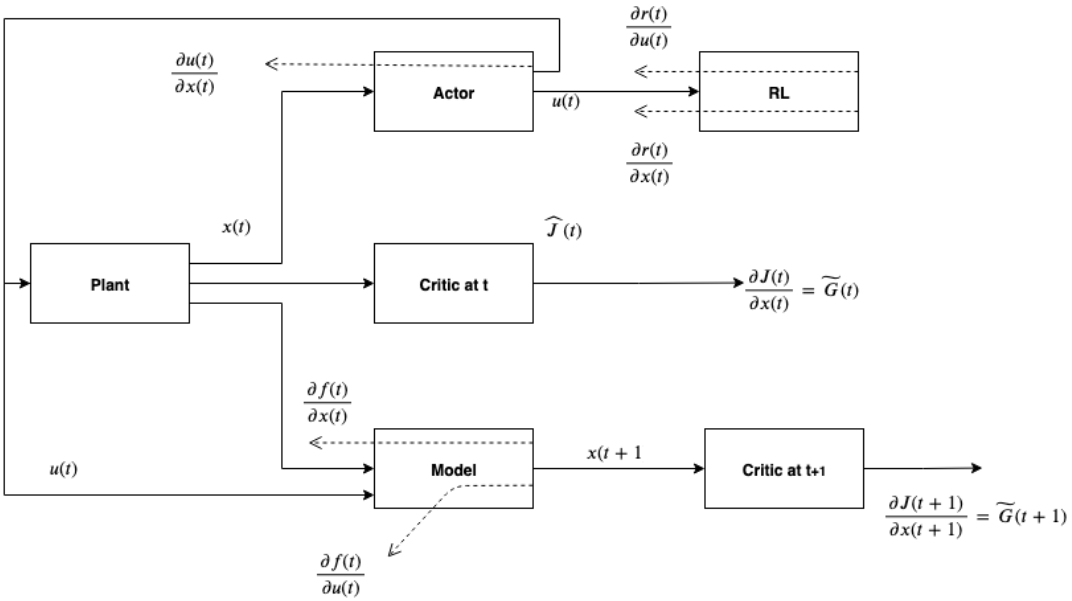


Figure (2.9) DHP scheme block diagrams and updates based on (Sutton and Barto, 2018)

2.5.4. Global Dual Heuristic Dynamic Programming (GDHDP)

If both the value function and its derivative are approximated in a Dual Heuristic Programming framework, the resulting architecture is called Global Dual Heuristic Programming (GDHDP). GDHDP will combine the advantage of both HDP and DHP and its motivated to result in faster convergence while decreasing the complexity of the function approximation (Stengel and Ferrari, 2004).

2.5.5. Comparison of different ACDs

Different researchers have compared ACD methods in aerospace fields and also power systems. Prokhorov *et al.* (1995) compared GDHDP, DHP and HDP controllers in auto-landing. Although it is concluded in this study that GDHD and DHP methods outperform HDP based on the success ratio, they might not be the best platform for fast online adaptation due to their need for more training time. For example, van Kampen *et al.* (2006) used an HDP approach for F-16 longitudinal control in an attempt to reduce the complexity and therefore faster training. He compared the performance of the offline trained HDP and ADHDP methods in adapting to online changes in the General Dynamics F-16. The experiments showed that while both methods are robust to some changes in the plant dynamics, HDP approach adapts better to the changes in the dynamics while being more sensitive to measurement noises. However, HDP can be used for more initial flight conditions (van Kampen *et al.*, 2006). One advantage of the ADHDP methods is that they eliminate the model approximation, but this does not necessarily make the problem simpler, and it also demands more training duration since the plant model is implicitly model in the parameters of the critic.

Van Kampen *et al.* (2006) also performed a comparison between HDP and ADHDP methods. In the offline learning phase, the HDP controller shows a higher chance of convergence to what is assumed as the control behaviour. In addition, almost a 100 percent higher success ratio is achieved for the HDP controller when the number of trials is kept constant. In the online learning phase, where the adaptation of the learned agent is tested, the HDP controller performs better in adaptation to the changes in the plant dynamics, when compared to the ADHDP controller both when the accuracy and the convergence rate is measured. However, the ADHDP controller behaviour in the existence of the noise in the system is better than the HDP. At the end of this study, it is shown that HDP can be used for a wider range of flight conditions if trimmed flight condition is assumed. More literature is available for comparison of ACD methods such as Stengel and Ferrari (2004), Venayagamoorthy *et al.* (2002) and etc.

2.5.6. Single Network Adaptive Critic (SNAC)

To decrease the complexity in actor-critic solution methods caused by the interaction of different NN approximations, an architecture which eliminates one of the function approximation in the AC framework proposed, called the Cost-function-based Single Network Adaptive Critic (J-SNAC). The schematic of the network training for a J-SNAC network is shown in Figure 2.10.

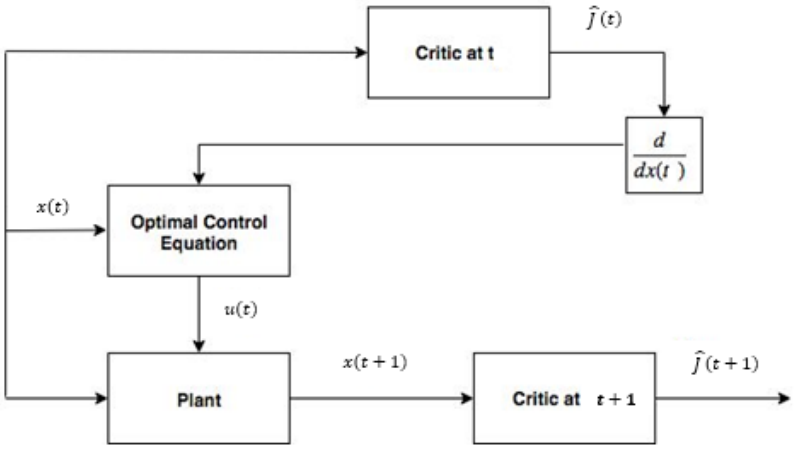


Figure (2.10) J-SNAC scheme block diagrams and updates based on (Ding and Balakrishnan, 2011)

The method first provoked by Ding and Balakrishnan (2011) which is also compared to a DHP method. The result shows the outperforming of the J-SNAC method in computational cost while achieving the same accuracy as DHP. Some recent studies including Modares *et al.* (2013), Kiumarsi and Lewis (Kiumarsi and Lewis), Kiumarsi *et al.* (2016) and Modares *et al.* (2014) are discussed the approach of having an explicit optimal control equation in off-policy methods and within actor-critic frameworks.

2.6. Safety in Reinforcement Learning

To make RL control methods more suitable for real-world applications, the system should be assured not to become unstable as the damages can be really fatigue, especially in automated flight control systems. Perkins and Barto (2002) and Balakrishnan *et al.* (2008) have discussed DP/RL approaches that will guarantee the stability of the system within the Lyapunov framework. In such RL approaches, called Stable Reinforcement Learning (StaRL), the stability of the system is guaranteed while the system moves in the direction of minimizing the cost function. Another way of looking at the safety is by considering the limitations on the affine in the nonlinear input systems where there are constraints imposed on the input signal. An example is the saturation and dead-zones in the actuators as well as the backlash. Most of the RL methods do not include such constraints in solving the HJB equation. Ignoring the actuators constraints can also result in instability of the system (Yang *et al.*, 2014).

As discussed in section 3.4 Lyapunov methods are widely used in controller design for objectives of stabilization or keeping the system's state in a defined operating range. Lyapunov functions in control theory field, are mathematical tools for analyzing the dynamical system stability. The integration of the Lyapunov methods with RL would propose a reliable RL controller that follow Lyapunov principles. In such an approach, the agent would learn only by exploring the given and base-level control actions that result in stability. Therefore, the chosen policies are assured to be safe. In an experiment by Perkins and Barto (2002) on a pendulum swing-up task aa cost-to-target problem with a target region, it is shown that some practical benefits such as basic performance guarantees and faster learning can be achieved. Such an outcome can be expected since the space that the agent is exploring is limited by imposing constraints on the control choices and therefore, some of the features of the RL methods such as exploration, convergence and FA parameters will not influence the stability of the system.

Fjerdingen and Kyrkjebø (2011) also proposed for a safe learning strategy with Lyapunov stability properties. In this study, the Continuous Actor-Critic learning Automation (CACLA) which is a combination of RL with control Lyapunov functions are used to assure the operation of the RL controller in the stability region of the system. It is shown that such a controller is able to find a good control solution comparable with discrete methods while the need for heuristic search is eliminated. Berkenkamp *et al.* (2017) believes that the fact that the RL agent should desirably explore all the possible actions can be harmful to real-world systems and that will limit the use of RL methods with systems that are safety critical. This study also performs an experiment on inverted pendulum problem, which ensures that the pendulum would not fall while the NN policy FA parameters will be optimized using the statistical model of the dynamics and Lyapunov stability verification that assure stability.

Another researcher who studied the safety with RL is Chow *et al.* (2018) who proposed a constrained Markov Decision Problems (CMDPs) which constraint the cumulative costs with Lyapunov methods. The fuzzy-based Lyapunov controller with only one function approximation for Q-function is discussed in Kumar and Sharma (2017). The study shows for an inverted pendulum balancing problem; the fuzzy-based Lyapunov controller reaches the highest success probability defined as the averaging the number of successful trials each consisting of 3000-time steps over the 50 number of episodes. The result shows that the fuzzy-based Lyapunov controller has the highest success probability of 99.6% the Lyapunov actor citric set-up almost 98 % and the conventional fuzzy Q-learning approach 91%.The AC controller train the two FAs with stability guarantee by updating the NN weights based on Lyapunov theory (Modares *et al.*, 2013). While the mentioned controllers all ended up with a swaying trajectory around the balancing point, the fuzzy Lyapunov controller is able to stay steady at the top vertical position of the pendulum. The same sequence of results holds for the controller performance by adding noise to the torque. The downside is that continuous actions cannot be considered for this approach.

Finally, Yang *et al.* (2014) developed a critic-only approach for online optimal adaptive control of the constrained systems by proposing a new critic update rule. This update rule ensures convergence of the critic network to optimal control and guarantees the stability of the closed-loop system and includes the minimization error parameters and Lyapunov's condition for stability which result in optimality without policy iteration.

2.7. Conclusion

In this chapter, the topics associated with Reinforcement learning problem in general with a focus on the continuous/approximate application were discussed. First, a description of the RL field and its history is given. Then the RL problem, elements and solutions were discussed. Different classification for RL problems are then illustrated, followed by an explanation on one of the most widely used RL approaches adaptive critic designs. Finally, the safety in RL solutions as one of the steps toward practicality is mentioned.

The methods that are used to solve a finite MDP can be described in three fundamental classes of Dynamic Programming (DP), Monte Carlo (MC) methods, and Temporal-Difference (TD) learning, each has their pros and cons. DP methods, in contrast to TD, require an accurate model of the environment, although they are mathematically well-developed. MC methods are the simplest; however, they are not suitable for incremental and model-free approaches. Instead, TD approaches are sophisticated and fully incremental. These methods can be combined to achieve higher efficiency in convergence rate, e.g. by using eligibility traces or the combination of DP and TD, which is possible within approximate solutions. To reach optimality, the importance of exploration and its trade-off with exploitation should not be ignored. The smart design of the reward-function also plays an essential role in convergence to the correct optimal policy.

The solution methods used to solve diverse formulation of the RL problem can be classified according to the latest release by Sutton and Barto (2018). These are the tabular solution methods in which the state and action spaces are small enough or can be presented as discrete or the value-function can be given in tables or arrays. The other category is approximate solution methods that use generalization methods and can be applied to much larger problems. Approximate methods are extensions of tabular methods to apply for large state spaces. The critical issue in approximation methods is the generalization between the states that are already visited with those that have never been encountered. This generalization can be achieved with Function Approximation (FA) methods. Main challenges for RL with FA include bootstrapping, non-stationarity and delayed target. Therefore, in approximate RL, prohibitive memory and computation costs should be avoided, especially in online case. Also, in learning, generally a limited amount of data is available. Therefore, the power of the approximator can change the amount of data required for training and the solution accuracy that can limit the resolution for a given data. The training for approximate approaches can be done with two methods of on-policy and off-policy. For on-policy training, two paths exist, the prediction case in which for a given policy the value-function is approximated and the control case, for which the optimal policy is approximated.

Another formulations of the RL problems can be based on their learning approach (active/passive, online/offline), the agent design (model-based, model-free) or the exploration routine (directed/undirected). The categorization of interest in this study is the classification which is based on the information that must be learned which divide into model-based and model-free. The notation can be interpreted differently, although both are the utility-based algorithms. As the name already indicates the main difference between them is that the model-based methods, such as PEGASUS, deals with a prior known probability model plus a value function for deriving the optimal policy. It solves for the optimal policy for a defined goal based on the Markov Decision Processes (MDPs). On the contrary, the model-free approaches, such as Q-learning and SARSA, learn values by trial-and-error and looks for the optimal overall values using the action-value function or Q-function. Switching to model-free approximate RL is not trivial as challenges such as memory intensiveness, the convergence-rate, the adaptability and their greedy nature that can limit exploration exist for such approaches.

Within RL approaches that can deal with continuous actions, critic-only methods may not be a good option, since deriving the control law require solving an optimization procedure. Only if the action space for the controller is small and finite, critic only methods can be useful, since the optimization can be done by enumeration. Actor-only methods, on another hand, can result in high variances in the gradients and are not suggested, although they can adapt faster to changing environments compared to critic-only algorithms. Actor-critic methods (also known as policy gradient methods), therefore, are emphasized more for non-stationary settings that continuous state and action spaces are required. They combine learning in policy space with value function approximations that result in algorithms which prove convergence on two certain time-scales. This structure has shown promising results for optimal adaptive control of non-linear systems by exploiting more complex FAs.

The Deep Learning RL methods are also discussed in this chapter due to their new attraction by RL scientists. Deep RL in pattern recognition will be employed when broad unlabeled or unstructured data is available and efficient categorization is needed. It should be noted that when Deep learning is used, more tuning will become necessary. This might arise more complexity to already complicated systems since more adjustment of the FA parameters will become required.

The adaptive critic design methods and single network adaptive critic approaches are also studied in this chapter due to their importance in RL control. Within ACD methods, action-dependent algorithms can be assumed model-free while others are model-learning methods that need to learn the underlying plant dynamics to come to the explicit approximation of the value function. Model-learning processes can be less complicated due to the separation of the tasks and will need less data for training. To decrease the complexity, even more, the SNAC approach is proposed which only limit the approximation to the critic and gives an explicit control definition for the control action based on the value-function estimate.

Finally safe reinforcement learning methods are discussed. In summary, Lyapunov reinforcement learning constrained and limit the available actions to the RL controller to those which ensure stability by ensuring that the Lyapunov function value would be negative. An attempt to make the Lyapunov function to its maximum negative value would result in the optimal policy that also ensures stability (Lyapunov constrained action set).

More elaboration on Reinforcement Learning methods can be found in the book of Sutton and Barto (1998, 2011) with most recent updates in the draft book from same authors (Sutton and Barto, 2018). For the latest implementations of RL in Artificial Intelligence (AI) and a more general overview, the reader is referred to Russell and Norvig (2016). The book of Busoniu *et al.* (2010) provides a thorough overview of continuous and approximate RL methods. For optimization of control, RL approaches of Gosavi (2003) give a good background on the techniques.

3

Reinforcement Learning-based Flight Control System Design

As it was proposed in the introduction, continuous reinforcement learning methods have attractive qualities of being model-free processes applicable to nonlinear and time-varying systems. Dynamic adaptability, the generalization of abundant resources, and mathematical explicitness of the optimization problem are other features of an approximate RL controller. However, Reinforcement Learning (RL) is a general problem that has been studied more in the field of Artificial Intelligence (AI) and has its roots in computer science. Although RL was already a subject of interest to researchers in the 20th century, it was only introduced in control problems by the mid-1950s. Since the aim of this thesis project is to employ RL for the design of an optimal and adaptive controller; therefore, in this chapter, these concepts are dealt with. The chapter starts with an introduction in section 3.1 and a summary background on adaptive and optimal control in section 3.2 and section 3.3, mostly using RL language. The sections include a discussion on how these control objectives can be achieved by RL-based control i.e., the place of RL in adaptive and optimal control. A hybrid approach of safe reinforcement learning, which combines reinforcement learning with Lyapunov-based stability control, is introduced in section 3.4. The chapter concludes with the applications of RL in flight control system designs in section 3.5 with the conclusion in section 3.6. The following terms are used interchangeably between AI language and the control engineering literature.: ”agent” and ”controller”, ”system” and ”environment”, ”learning rate” and ”tuning gains” and ”action” and ”control signal”.

3.1. Introduction

From the system control perspective, control problems divided into two categories: a tracking problem with the objective of following a reference trajectory and an optimal control problem, in which the objective is the optimization of a controlled system's behavior measure (which is not necessarily the tracking error) (Sutton *et al.*, 1992). The adaptive optimal control problem is defined as the ability to learn the optimal control solution online for uncertain systems (Bhasin, 2011). RL can be seen as a direct approach to adaptive optimal control for non-linear systems, an approach where control rules are determined without access to an explicitly controlled system model (Sutton *et al.*, 1992). In other words, RL is a bridging between conventional optimal control, adaptive control and bio-inspired learning techniques (Khan *et al.*, 2012). Following the research problem, in this chapter, the recent developments in each of these control aspects are studied, and their relevancy to continuous RL approaches are presented.

In general, the design of any flight controller requires a model construction with parameters estimated through system identification of the flight test data. The control approach then will be applied to the linearized model (directly or incrementally) or in more complicated cases straightly to the nonlinear model. For the linearized model, several control strategies of the PID controller, H_∞ controller, and Linear Quadratic Regulator (LQR) controllers can be applied. However, these approaches come

with problems of model reduction, accuracy, modeling errors, and tuning of the gains (PID) and weight matrices (LQR and H_∞). One of the main characteristics of all these approaches is that they require the desired controller output to be instructed to them. By the introduction of RL to control theory, the controller is now responsible for reaching the optimal control action by achieving a defined target cost-function. Therefore, in this chapter, the main techniques for combining RL with optimal and adaptive control are presented through literature. Thus, the chapter does not include detailed documentation on RL, but it highlights the research on RL with particular relation to the control aspect. For more discussion on RL in particular, the reader is referred to chapter 2.

Optimal control corresponds to a group of methods that are used to generate the best control policy to result in the best possible predefined criterion, e.g. maximum performance. Normally, the solutions to optimal control problems, such as Pontryagin's minimum principle and HJB optimallity require complete knowledge of the system dynamics (Vrabie and Lewis, 2009). An example of researchers work in parallel on similar approaches to RL but in optimal control, e.g. methods such as Adaptive Dynamic Programming (ADP) and Neuro-dynamic programming (NDP), are Werbos and Bertsekas. However, these approaches later combined with RL in studies by Sutton, Werbos in nineties and recently by (Sutton *et al.* (1992), Sutton and Barto (1998), Williams (2009), Lewis *et al.* (2012), Vrabie and Lewis (2009), Werbos (1992)). Interesting surveys on dynamic programming and ADP/NDP can be found in Bertsekas *et al.* (1995) and Bertsekas (2007). In the following some of these literature are highlighted. The model-free nature of the RL is not separately discussed as a section in this chapter, however, it is mentioned in the context of adaptive and optimal control with RL.

3.2. Reinforcement Learning and Optimal Control

The term "Optimal Control" was introduced in the late 1950s to refer to the problem of optimizing a dynamical system's behavior functional over time. Two approaches were taken, one by Richard Bellman in the mid-1950s and the other by an extension of Hamilton and Jacobi's theory. When talking about optimal control, it should be specified regarding what the optimality needs to be achieved? In other words, how the cost-function and performance measures are defined? Another notion that needs to be defined is that in what time span the optimization should take place? Is it for a defined period (finite-horizon) or over the whole time of the system operation (infinite-horizon)? One of the first talks on optimal adaptive control and RL was by Sutton *et al.* (1992), who referred to RL as a direct adaptive optimal control approach. Most of the literature available for adaptive optimal control is through a modern formulation of RL called approximate dynamic programming (ADP). Within this category, adaptive actor-critic methods are more emphasized. RL methods in control context of single-agent systems are mainly used to solve optimal regulation and tracking problems (Kiumarsi *et al.*, 2018). In optimal regulation problem, the objective is to assure the convergence of states and outputs to a defined desired value, while in an optimal tracking control problem, the goal is that the states or outputs keep tracking a reference trajectory.

Khan *et al.* (2012) refer to actor-critic methods as the most popular and effective method between reinforcement learning approaches for optimal adaptive control, also presented in the work of Vrabie and Lewis (2009) and Al-Tamimi *et al.* (2007). Most recent studies by Lewis et.al presents reinforcement learning in the context of ADP. This includes online AC algorithms for infinite-horizon optimal control in continues time (Vamvoudakis and Lewis (2010), Khan *et al.* (2012)) and optimal adaptive control for unknown systems (Lewis and Vamvoudakis, 2011). NN function approximations are used for both the actor and the critic estimations Stingu and Lewis [125]. The structure would eventually result in an estimation of the optimal control policy. Modares et.al. studied continuous reinforcement learning and feedback control for unknown constrained-input systems with policy iteration and actor-critic structures (Modares *et al.* (2013), Modares *et al.* (2014)). He proposed integral reinforcement learning with experience replay for partially unknown systems improving the speed of learning and the overall performance of RL controller for continuous time systems. A survey of recent developments in autonomous RL-based optimal feedback tracking control is given by Kiumarsi *et al.* (2018).

In general ADP methods which use adaptive critic structures (value-function approximation) and, there-

fore, are known as adaptive critic designs (ACD) have strongly focused on in RL-based optimal control community (Khan *et al.*, 2012). These methods, which are under actor-critic TD methods, are explained in detail in chapter 2 with classifications, block diagrams and update rules. The stability of these controllers and the model-free and model-learning approaches are discussed in the work of Balakrishnan *et al.* (2008). In addition, Vrabie and Lewis (2009) proposed a ACD for adaptive optimal control of an unknown nonlinear continuous-time systems, using policy iteration and NN approximation. The convergence proof is provided for this scheme with stabilizing control policy initialization. Unlike traditional DPs, the main directions for recent studies on ADP for optimal adaptive control are focused on online and model-free approaches that can carry out time-varying dynamics.

Optimality, in the flight control sense, can be interpreted as the minimum tracking error and/or minimum use of energy or in other words, minimum physical interaction incorporated in the cost function. Safety constraints can be added to this cost function in terms of actuator limits, e.g., by integrating Lyapunov theory to RL control or by simply saturating the outputs of the controller. Although RL has different applications in many fields, many of the developments in, e.g., robotics control, can also be applicable to flight controller designs. For example, one of the most recent works in RL control is the use of experience replay (Adam *et al.*, 2012) which use online learned data to the RL agent repeatedly or using trust regions for the optimization problem (Schulman *et al.*, 2015). These methods, although been tested in robotics-related simulation and some cases practiced in real-life experiments, there is no limit to employ them for flight control simulations that concern with optimality desire. In addition, the reinforcement learning extended less than two decades ago to systems with continuous state and action spaces by the work of Doya (2000).

3.2.1. Optimization methods and function approximation

Optimization techniques play an important part in the engineering process. Many real-world systems are highly uncertain and non-linear, so deriving an optimal solution to a given problem using purely analytical means is not always possible. Therefore, a variety of tools is applied to solve these types of problems: linear programming, integer programming, and heuristic methods, among many others. RL, on the other hand, is closely related to classical optimal control, in the sense that both problems deal with finding an optimal policy by optimizing an objective function for a system described by state-space (Kober *et al.*, 2013). The performance of an optimal controller is highly dependent on the control laws used for translating sensor measurements to control surface outputs, i.e., the Control Allocation (CA) methods. For a redundant actuation aircraft, the number of control inputs is larger than the number of control variables, and hence, a suitable CA method is required to attain a control command with the optimal utilization of the available control effectors. For an RL controller, there is no desired control command. Hence, optimal control with RL can be seen as a constrained optimization problem with the classical cost-function. To introduce control allocation, the cost function can include the control surfaces use as a parameter that needs to be optimized. This problem has been studied in several kinds of literature, some are Zhou (2018), Ferrari and Stengel (2004) and Sutton *et al.* (1992).

RL can be seen as a repetitive process of learning parameter optimization. Most of the optimization methods used in optimal control are theory-based. In these approaches, the heart of the design of an optimal controller (e.g., for nonlinear systems) is solving the Bellman optimality equation (also known as Hamilton-Jacobi-Bellman (HJB) equation). For a continuously differentiable approximated value-function, Bellman's principle of optimality can be used to solve the optimization problem. The optimality condition, in this case, is a nonlinear partial differential equation (PDE), also known as the HJB equation (Bhasin, 2011) that can result in the optimal control policy. However, finding an analytical solution to the HJB equation is difficult and sometimes even not possible. One numerical approach to this problem is the approximation of state-space and actions with FAs, and then solve the nonlinear optimization problem using methods such as pseudo-spectral and direct shooting (Betts and Kolmanovsky, 2002). However, these methods are offline, model-based, and sensitive to initial conditions. RL policy iteration methods are used for approximation of the HJB equations and algebraic Riccati equation by carrying out successive iterations. These methods require complete knowledge of the system dynamics (Kiumarsi *et al.*, 2018). In some RL approaches, e.g. van Kampen *et al.* (2006), the optimization is based on the designer experience and a prior information is given to the controller

before online learning.

Methods that use feedback linearization such as Differential Riccati Equation (DRE) (Freeman and Kokotovic, 1995) have the downside of omitting part of the system which is non-linear, while this information might be useful to the system. A recently developed robust identification-based state derivative estimator based on ACDs, which identifies the system model with two neural network architectures for approximating the actor and the critic, is an online control approach for continuous RL (Bhasin, 2011). This indirect adaptive method is only partially model-free and learns the optimal solution for infinite-horizon optimal control. Hence, convergence to optimal control is an ongoing challenge for RL methods dealing with continuous state and space systems. However, still, the promising approaches such as the combination of RL with NN or other memory structured methods can eliminate the need for a full knowledge of the system to some good extent. These structures are divided into two categories of actor-critic (AC) architectures and Q-learning. Q-learning is a simpler approach to discrete-time systems, although the recently developed Integral Q-learning extends its use to continuous systems (Lee *et al.*, 2012). The AC designs that use back-propagation algorithms and ANN and their application for continuous-time systems has been widely explored in Bhasin (2011), Van Kampen *et al.* (2006), van Kampen *et al.* (2006), Zhou *et al.* (2017), Zhou *et al.* (2018). AC methods approximate the solution of the HJB equation and eliminate the requirement for a complete knowledge of the system. In AC structure, the critic is updated in the direction of minimizing the Bellman error while the actor is updated to minimize the value function (Bertsekas *et al.* (1995), Werbos (1992)). It will be shown in chapter 2 that this structure does not need complete knowledge of the system dynamics.

For discrete-time systems, the synchronous and online methods for actor and the critic updates are presented in He and Jagannathan (2007) and Dierks and Jagannathan (2011). Luo and Zhang (2008) introduced the greedy iterative heuristic DP (referred to as HDP and its categories in chapter 2), which add another neural network FA for approximating the unknown plant dynamics. The plant approximation make these methods independent of a prior model of the system and categorize them as model-learning methods (Grondman, 2015). LQR optimal control with reinforcement learning can be seen in the work of Vrabie *et al.* (2008), where policy iteration methods are employed to solve discrete-time LQR problem.

In general in AC methods, and in order to find the action which results in the optimal value-function in the presence of the continuous actions, function optimization techniques need to be applied for two cased, minimization of the Bellman error (also known as TD error) and the value function (if it is defined with a cost function). This makes the choice of the value function approximation very critical for convergence and performance of the RL algorithm since the option for applying optimization methodologies is depending on the type of the generalization function used for storing the actor and critic networks. For linear systems, converging to the optimal global solution is guaranteed due to the quadratic structure of the value function estimation. This is not the case for nonlinear systems.

Some of the optimization methods include the Particle Swarm Optimization (PSO) algorithm, which applies to many types of systems and functions, including discrete and non-differentiable functions (such as Cerebellar Model Articulation Controller (CMAC) networks). Numerical optimization methods such as Newton-Raphson or wire-fitting might be used for optimization Q-value estimations. Other methods include gradients descent optimization based on local function approximations of the value function and the policy used in a particular subset of RL algorithms, denoted as AC methods (Fjerdingen and Kyrkjebø, 2011). Using exponential-based Radial Basis Functions (RBFs) of NNs has a significant advantage over the use of CMAC since RBFs activation functions are twice differentiable, and optimization methods are available for them. Another choice for FAs is the polynomial functions, which are more computationally efficient and capable of finding the global maxima/minima of the function. However, for neural networks output optimization, RBF-based function approximation is more beneficial than polynomials functions due to the closer approximation of the network (Naruta, 2017). Lyapunov optimization methods have also been used for dynamical systems for stability guarantees. The Lyapunov approach, however, depends strongly on the Lyapunov function description, defined by expert knowledge.

3.2.2. Optimal tracking control

The main objective of this thesis project, also part of the research questions 3 and 4, is to present the capabilities of the proposed approximate RL control approach in a suitable control task. To achieve this goal, the result of the controller will be compared to the discrete RL implementation for the ICE model in a reference tracking control task, already available in literature by de Vries (2017). Therefore, in this section, the optimal tracking problem is discussed for a general RL-based optimal control problem considering actor-critic structure that described in chapter 2. It should be noted that no specific approach of ADP is aimed at here. But it is tried to gather a general framework for optimal tracking control with RL. Consider the dynamics of the nonlinear system with perfect measurements with affine input given as:

$$\begin{aligned} \dot{x}(t) &= f(x,t) + g(x,t)u(t) \\ y(t) &= l(x,t) \end{aligned} \tag{3.1}$$

Where $x \in \mathbb{R}^n$ is the state vector of the system, $u \in \mathbb{R}^m$ is the control input, and $y \in \mathbb{R}^p$ is the system output in t time. $f(x,t) \in \mathbb{R}^n$ is the drift dynamics function, $g(x,t) \in \mathbb{R}^{n\times m}$ is the input dynamics, and $l(x,t)$ is the output (observation) functions. It is assumed here that the system dynamics are effected by the control input $u(t)$. Control inputs for flight control systems are usually corresponds to forced and moments induced by control effectors, described in the form of:

$$u(t) = \pi(x,t) \tag{3.2}$$

and

$$|u_j(t)| \leq \bar{u}_j, \qquad j = 1,\dots,m \tag{3.3}$$

Where π is the policy function, $u(t) \in \mathbb{R}^m$ and $\bar{u}_j$ is the saturating bound for the jth actuator. For an optimal tracking controller, the goal is to derive the optimal control input which results in the states of the system $x(t)$ to follow the desired reference trajectory signal $x_d(t)$ (for the regulation problem $x_d(t)$ equals a constant value for all times). For such a controller, the measure of performance is defined as the tracking error $e_t(t)$ given by:

$$e_t(t) = x(t) - x_d(t). \tag{3.4}$$

Where $x_d(t) \in \mathbb{R}^n$ can be assumed to be generated with a command generator model as:

$$\dot{x}_d(t) = \mu_d(x_d,t) \tag{3.5}$$

Since the control law depends on the states and as well the trajectory error, the augmented system dynamics with error and reference trajectory dynamics and augmented state of $X(t) = [e(t)x_d(t)]^T$ can be given as:

$$\begin{aligned} \begin{bmatrix} e(t+1) \\ x_d(t+1) \end{bmatrix} &= \begin{bmatrix} f(e(t)+x_d(t)) - \mu_d(x_d(t)) \\ \mu_d(x_d(t)) \end{bmatrix} + \begin{bmatrix} g(e(t)+x_d(t)) \\ 0 \end{bmatrix} u(t) \\ &\equiv F(X(t)) + G(X(t))u(t) \end{aligned} \tag{3.6}$$

The RL-based optimal tracking control method minimize the system cost/utility function while getting closer to the goal. The cost function with a quadratic criterion is defined by:

$$U(e(t),u(t)) = e_t(t))^T Q e(t) + u^T(t) R u(t) \tag{3.7}$$

Where Q and R are positive deterministic matrices for the tracking error and control inputs utilization.

The cost-to-go function, considering discounted reward/cost, is defined by:

$$J(x(0), x_d(0), u(t)) = \sum_{i=0}^{\infty} \gamma^{i-t}[(x(t) - x_d(t))^T Q(x(t) - x_d(t)) + u^T(t)Ru(t))] \tag{3.8}$$

With $Q \succeq 0$ and $R = R^T > 0$ are the components of the cost/utility function and γ is the discount factor. The value-function, which is the cumulative sum of future cost from the current state and policy π, is defined in terms of the systems states as:

$$\begin{aligned} V(X(t)) = V(e(t), u(t)) &= \sum_{i=t}^{\infty} \gamma^{i-t} U(e, u, t) \\ &= \sum_{i=t}^{\infty} \gamma^{i-t}(e^T Q e^T + u^T R u) \end{aligned} \tag{3.9}$$

The control input for a tracking problem here consists of only a feed forward term $u_a(t)$, which excite tracking with the approximated control input that will be improved by back-propagation from the actor network for tracking.

$$\hat{u}(t) = u_a(t) \tag{3.10}$$

As one can see from Equation 3.10 the control input requires knowledge of the system dynamics and the dynamics of the trajectory. Conventional methods exist which derive the control input based on a complete understanding of the system dynamics. In Kiumarsi and Lewis (2015), a formulation based on actor-critic approaches is proposed that give a comprehensive definition for optimal control input and, therefore, abandoned the prior knowledge about the system while providing an explicit definition for the optimal control input. This approach, however, is tested for problems with no more than two states and one control input. In this study, the Equation 3.9 can be rewritten to the form of:

$$V(X(t_k)) = V(e(t), u(t)) = U(e(t), u(t)) + \gamma V(x(t+1)) \tag{3.11}$$

The Bellman equation for this problem is given as:

$$H(x(t), u(t), V) = x^T(t)Qx(t) + u^T(t)Ru(t) + \gamma V(x(t+1)) - V(x(t)) \tag{3.12}$$

Based on the Hamiltonian equation, the optimal value function (the HJB equation) is given by:

$$V^*(X(t)) = min_u[U(e(t), u(t)) + \gamma V^*(x(t+1))] \tag{3.13}$$

The optimal control it then obtained for the such a system as:

$$\begin{aligned} u^*(X(t)) &= argmin_u[V^*(x(t))] \\ &= -\frac{\gamma}{2} R^{-1} G^T(x)\left(\frac{\partial V^*(x(t+1))}{\partial x(t+1)}\right) \end{aligned} \tag{3.14}$$

In the next section, the actor and critic networks and their rule updates are described for a generic actor-critic architecture with more than one control power available for the controller.

A. Actor-Critic Network

Since perfect measurements are assumed, no optimal estimator for states is required. To abandoned the knowledge of the dynamics, an actor-critic scheme is defined for the approximation of the HJB equation. The critic and actor networks update the value function and policy simultaneously (in contrast to online policy iteration algorithms in which the updates happen sequentially). It is assumed that both networks are approximated with two-layer NN with one hidden layer. If the model of the plant needs to be estimated, an additional model network will be added to the framework that will follow the same update rule as a critic, however, with the error defined as the difference between the target state values and the output of the network.

A.1 Critic Network

The input for critic network is the assumed augmented state $X(t)$ and the output is the estimated value function $\hat{V}(X(t))$. The network is defined then:

$$\begin{aligned}\hat{V}(X(t)) &= W_c^o(t)\phi_c(W_c^h X(t)) \\ &= \sum_{i=1}^{n_c} \hat{w}_{ci}^o(t)\phi_{ci}(\sum_{j=1}^{2N} \hat{w}_{cij}^h(t)X_j(t))\end{aligned} \tag{3.15}$$

Where $\phi_c(t) = [\phi_{c1}(t), ..., \phi_{cn_c}(t)]^T \in \mathbb{R}^{n_c \times 1}$ is the hidden layer activation function vector (e.g. tanh), W_c^h is the output layer weights defined by as:

$$W_c^h = \begin{bmatrix} w_{c11}^h & \cdots & w_{c1(N)}^h \\ \vdots & \vdots & \vdots \\ w_{cn_c1}^h & \cdots & w_{cn_c(N)}^h \end{bmatrix} \in \mathbb{R}^{n_c \times N} \tag{3.16}$$

where n_c and N are the number of the neurons for the hidden layer and the number of inputs, respectively. w_{cij}^h denotes the weight from the jth input value to the ith hidden neuron. The output layer weights are defined as:

$$W_c^o = [w_{c1}^o, w_{c2}^o, ..., w_{cn_c}^o] \in \mathbb{R}^{1 \times n_c} \tag{3.17}$$

where w_{ci}^o is the weight from ith hidden layer and output layer.

A.2 Actor Network

This NN approximates the control input for the nonlinear tracking problem. The data to the actor network is $X(t)$ as well, and the output is the control command given as:

$$\hat{u}(X(t), W_a^h(t), W_a^o(t)) = [\hat{u}_1, ..., \hat{u}_m] \tag{3.18}$$

where

$$\begin{aligned}\hat{u}_i &= \sigma_i(W_a^o(t)_{ai}\phi_a(W_a^h(t)X(t))) \\ &= \sigma_i(\sum_{k=1}^{n_a} \hat{w}_{a_{i_k}}^o(t)\phi_{a_k}(\sum_{j=1}^{N} \hat{w}_{a_{i_j}}^h(t)X_j(t)))\end{aligned} \tag{3.19}$$

where $\phi_a(k) = [\phi_{ak}(t), ..., \phi_{a_{n_a}}(t)]^T \in \mathbb{R}^{n_a \times 1}$ is the hidden layer activation function vector and σ is the basis function for the output layer, n_a is the number of neurons for the hidden layer and $i = 1, ..., m$ is

the ith output-layer. The weights matrix for hidden layer is defined as:

$$W_a^h = \begin{bmatrix} w_{a11}^h & \cdots & w_{a1(N)}^h \\ . & . & . \\ . & . & . \\ . & . & . \\ w_{an_a1}^h & \cdots & w_{an_a(N)}^h \end{bmatrix} \in \mathbb{R}^{n_a \times N} \tag{3.20}$$

where $w_{a_{ij}}^h$ is the weight connecting jth input to the ith hidden neurons. The matrix of the weights between the hidden and output layer is defined as:

$$W_a^o = \begin{bmatrix} w_{a11}^o & \cdots & w_{a1(N)}^o \\ . & . & . \\ . & . & . \\ . & . & . \\ w_{an_a1}^o & \cdots & w_{an_a(N)}^o \end{bmatrix} \in \mathbb{R}^{n_a \times N} \tag{3.21}$$

where $w_{a_{ij}}^o$ is the weight relating jth hidden layer to ith output layer.

B. Training the Networks

The gradient descent rules are used for updating the critic and actor network weights. The direction for the critic network is toward minimizing bellman error, and for the actor, the objective is to reduce the value function estimation (if J^* is zero). In the following, the update rules for the actor and the critic are illustrated separately.

B.1 Critic Training

If the Bellman prediction error is defined as:

$$e_c(t) = U(t) + \gamma \hat{V}(X(t)) - \hat{V}(X(t-1)) \tag{3.22}$$

where the reinforcement signal is defined by:

$$U(t) = X(t)^T Q X(t) + u(t)^T R u(t) \tag{3.23}$$

The Bellman error is defined as:

$$E_c(t) = \frac{1}{2} {e^2}_c(t) \tag{3.24}$$

The NN weights are tuned by the gradient descent algorithm as:

$$\widehat{W}_{t+1} = \widehat{W}_t + \Delta W, \qquad \Delta W = -\alpha \frac{\partial E}{\partial W_t} \tag{3.25}$$

where α is the learning rates. For the critic network which approximates the value function and is the function of the tracking error and the reference trajectory, the tuning laws apply as:

$$\hat{w}_{c_{ij}}^h(t+1) = \hat{w}_{c_{ij}}^h(t) - \alpha_c \frac{\partial E_c(t)}{\partial \hat{w}_{c_{ij}}^h(t)} \tag{3.26}$$

$$\hat{w}_{c_i}^o(t+1) = \hat{w}_{c_i}^o(t) - \alpha_c \frac{\partial E_c(t)}{\partial \hat{w}_{c_i}^o(t)} \tag{3.27}$$

Applying the chain rule for gradient descent approach for the hidden layer gives:

$$\begin{aligned} \frac{\partial E_c(t)}{\partial \hat{w}_{c\,ij}^h} &= \frac{\partial E_c(t)}{\partial e_c(t)} \frac{\partial e_c(t)}{\partial \hat{V}(t)} \frac{\partial \hat{V}(t)}{\partial \phi_{c_i}(t)} \frac{\partial \phi_{c_i}(t)}{\partial \hat{w}_{c\,ij}^h} \\ &= \gamma e_c(t) \hat{w}_{c_i}^o(t) (1 - {\phi^2}_{c_i}(t)) X_j(t) \end{aligned} \tag{3.28}$$

same for output layer gives:

$$\frac{\partial E_c(t)}{\partial \hat{w}^o_{c\,i}} = \frac{\partial E_c(t)}{\partial \hat{V}(t)} \frac{\partial \hat{V}(t)}{\partial \hat{w}^o_{c_i}(t)} = \gamma e_c(t) \phi_{c_i}(t) \tag{3.29}$$

The convergence for this update rule is not guaranteed due to the nature of the approximation with this gradient descent method.

B.2 Actor Training

The objective for the actor network is the minimization of the error between the approximated value function and the optimal value function. Therefore, a target control input is obtained by minimizing this error defined as the approximation error. To derive the control value at the next time step, in conventional control approaches, a model of the system is required. This model in some actor-critic structures such as ADHDP is approximated implicitly within the critic, and in other methods of HDP and DHP, then the model network will be to predict the future value of states. In another approach that abandoned the plant approximation, the previous values for the states are stored, and the error between the target and actual control inputs for the available current estimate of the actor and the critic are used for minimization. The stored states are then used as the actor and critic networks input. Therefore, the error for actor network is defined as:

$$e_a(t) = \tilde{V}^*(t) - \hat{V}(t) \tag{3.30}$$

The square actor NN is defined as:

$$E_a(t) = \frac{1}{2} e^2{}_a(t) \tag{3.31}$$

The same gradient descent tuning laws can be applied for the actor network as:

$$\hat{w}^h_{a_{ij}}(t+1) = \hat{w}^h_{a_{ikj}}(t) - \alpha_a \frac{\partial E_a(t)}{\partial \hat{w}^h_{a_{ij}}(t)} \tag{3.32}$$

$$\hat{w}^o_{a_{ij}}(t+1) = \hat{w}^o_{a_{ij}}(t) - \alpha_a \frac{\partial E_a(t)}{\partial \hat{w}^o_{a_{ij}}(t)} \tag{3.33}$$

where α_a is the actor learning rate. The chain rule for actor gradient descent algorithms is derived for hidden layer as (if model approximation is considered):

$$\frac{\partial E_a(t)}{\partial \hat{w}^h_{a_{ij}}(t)} = \frac{\partial E_a(t)}{\partial e_a(t)} \frac{\partial e_a(t)}{\partial \hat{V}(t)} \frac{\partial \hat{V}(t)}{\partial x(t)} \frac{\partial x(t)}{\partial u(t)} \frac{\partial u(t)}{\partial \hat{w}^h_{a_{ij}}(t)} \tag{3.34}$$

and for the output layer weights:

$$\frac{\partial E_a(t)}{\partial \hat{w}^o_{a_{ij}}(t)} = \frac{\partial E_a(t)}{\partial \hat{V}(t)} \frac{\partial \hat{V}(t)}{\partial \hat{w}^o_{a_{ij}}(t)} \tag{3.35}$$

The simultaneous update of the actor and the critic make convergence guarantees more difficult to prove. Kiumarsi *et al.* (2018) to ensure sufficient exploration within the states of the system and convergence to the global optimum, the control input signal should satisfy the persistence of excitation (PE) condition. This can be achieved by adding sinusoids of varying signals to the control input or other methods. convergence proof for the actor-critic network. The approach proposed here can be seen as a generic actor-critic approach in solving optimal tracking control problem (van Kampen *et al.* (2006), Modares *et al.* (2014), Kiumarsi *et al.* (2018)). The convergence of actor-critic methods are provided in Dierks and Jagannathan (2011) and Luo and Si (2013).

Actor-critic structures are employed for optimal control of the fixed-wing aircraft by Ferrari *et al.* (2008)

in its full flight envelop in two-phase of *pre-training/offline* and online training. In the offline phase of the training, the gradient of the NN controller is put equal to the linear gain matrices for each operating point. To be more specific, a proportional-Integrator (PI) controller gains are used as the expected output of the NN in offline mode. The incremental approach in this phase is used for system dynamics, assuming time-varying effects and small perturbations. In the online phase, the actor-critic with DHP structure is used with the first update based on the weights obtained in offline learning. A modified resilient back-propagation algorithm (RPROP), which assures a constant decrease of the network errors, is used at the onset of the training, which avoids going to significant changes at the beginning of the training. The stop criteria for offline training is based on the mean-squared measure of the network errors after at least three epoch es are completed. The framework was tested for nominal flight conditions and in the occurrence of failures caused by small and large variations in the parameters of the model. It was shown that the online adaptive controller performs much better than the robust offline trained controller in failure cases. Chandramohan *et al.* (2007) implement the same structure, now using DHP methods.

3.2.3. Model-free control

Model-free in the context of RL can be seen as what Russell and Norvig (2016) mentioned in his article, as a game that you start without knowing the rules until you face the losing situation after a couple of moves. The terminology model-free control can be explained in some literature using incremental approaches in which the system dynamics are described by an ultra-local equation, and therefore, the need for global modeling is abandoned (Fliess and Join, 2013). In some literature, RL is seen as a static map between the control policy and associated system performance, where a model of the system is not necessary and can be approximated through the performance critic network (Vrabie and Lewis, 2009).

In general, RL methods that use Temporal-Difference (TD) learning are independent of a model in learning the optimal policy. According to direct or indirect adaptive approaches, the system dynamics will be approximated offline or online along with the Bellman error formulation and value function (and also policy function in Q-learning methods) approximation. To divide tasks more evident in the RL control structure, the plant dynamics will be approximated during the training phase. These RL schemes are explained more in detail in chapter 2. Therefore, so far, model-free approaches for RL can be categorized into incremental methods and model-learning methods. What they both have in common is that a global system dynamics knowledge is not a prerequisite for the controller.

3.3. Reinforcement Learning and Adaptive Control

Up to this point, the following was discussed for RL: the optimal control for RL is defined within the acquirement of the optimal policy, which is the policy that maximizes the expected total reward (Russell and Norvig, 2016). Model-free control is defined as the situation were the agent has no knowledge about the environment (or the reward function) before learning. Another feature of the RL-based control is its adaptability. In the following, the relation between RL and adaptive control is discussed. Conventional control systems such as Proportion-Integration-Differentiation (PID) controllers have been used to control linear systems. These methods are not suitable for nonlinear systems that are usually the cases in the real world. The classical control theory can still be used to approximate linear systems. Adaptive control, on the other hand, can be used to handle the nonlinearities of the system by using function approximation such as fuzzy systems and neural networks (Xie *et al.*, 2012).

Schaal (1997) had put the relation between adaptive control and RL very nicely in his paper. He mentioned that reinforcement learning is usually distinguished from adaptive control in that the learning system can have rather general optimization objectives—not just, e.g., minimal tracking error—and is permitted to fail during the process of learning, while adaptive control emphasizes fast convergence without failure. Thus, RL resembles the way that humans and animals acquire new movement strategies, while adaptive control is a special case of learning control that fulfills stringent performance constraints, e.g., as needed in life-critical systems like airplanes. This emphasis on convergence with-

out failure in adaptive control often comes in the form of more assumptions and restrictions on the underlying system, compared to very few assumptions made in model-free RL (Fliess and Join, 2013). Lastly, in adaptive control, the defined tasks are not episodic; therefore e.g., tracking is defined for the infinite horizon in contrast to RL, which can be applied for a finite horizon or duration.

3.4. Lyapunov-based Control

Lyapunov-based control is discussed in a separate section due to its recent application in reinforcement learning-based controls, which concern with safeties. A more comprehensive elaboration on safe reinforcement learning can be found in chapter 2. Stability analysis for uncertain nonlinear systems with multiple equilibrium states is significantly complicated. Within the methods proposed in the literature for stability guarantee of the controllers, e.g., Lyapunov's approaches, the algorithms based on Lyapunov's stability theorem is most widely employed for stability establishment of, particularly nonlinear systems. The theory of A.M. Lyapunov for the stability of dynamical systems is proven to guarantee stability by identification of what is known as the Lyapunov function for the system. In general, Lyapunov functions can be seen as the generalized energy functions of the state of the system. The stability of the system is then assured by showing the continuous decrease of this function along the trajectory to the point of local minimum energy (Vincent and Grantham, 1997).

Different types of stability exist for dynamical systems. In the case of trajectory control, the system state may be required to: become close to a target point, remain close to some target point, or in some cases, enter a region of the target. By describing a dynamical system under control to be stable, it generally means that the system trajectory will reach and enter a target region in finite time with the probability of one (this is not necessarily a specified limited time) (Perkins and Barto, 2002). In control engineering, the Lyapunov theorem and its extensions are used to guarantee that the state of the system would reach the target. In this approach, if the state of the minimum energy in the Lyapunov's function is in the target area, it is guaranteed that the Lyapunov-based controller will bring the system to the target point. That is the reason that Lyapunov-based controllers are called stabilizing controllers. So far, apart from the Lyapunov theorem, there is no other general procedure that would guarantee the stability in control of nonlinear and arbitrary dynamical systems. However, the downside with Lyapunov analysis is that it requires complete knowledge of the system dynamics with the differential equations in some cases. Therefore, these methods are not straightforward.

Most of the controllers designed based on RL so far do not offer any guaranty on the stability of the controlled system. Lyapunov analysis has recently been employed in reinforcement learning control architecture by translating Lyapunov functions based on control constraints to ensure such stability. Although the freedom of the RL controller is decreased in these methods, these controllers are assumed to have considerable freedom in cost minimization comparing to conventional Lyapunov-based controller. This freedom or exploration feature is achieved by introducing RL to these controllers (Perkins and Barto, 2002). The most critical part of Lyapunov-based reinforcement learning is the appropriate choice of the Lyapunov function for designing a stable controller. In the work of Kumar and Sharma (2017), it is shown that defining the suitable Lyapunov function is not straightforward, e.g., for a swing-up pendulum task. Establishing a proper Lyapunov function will become more critical for more complex systems.

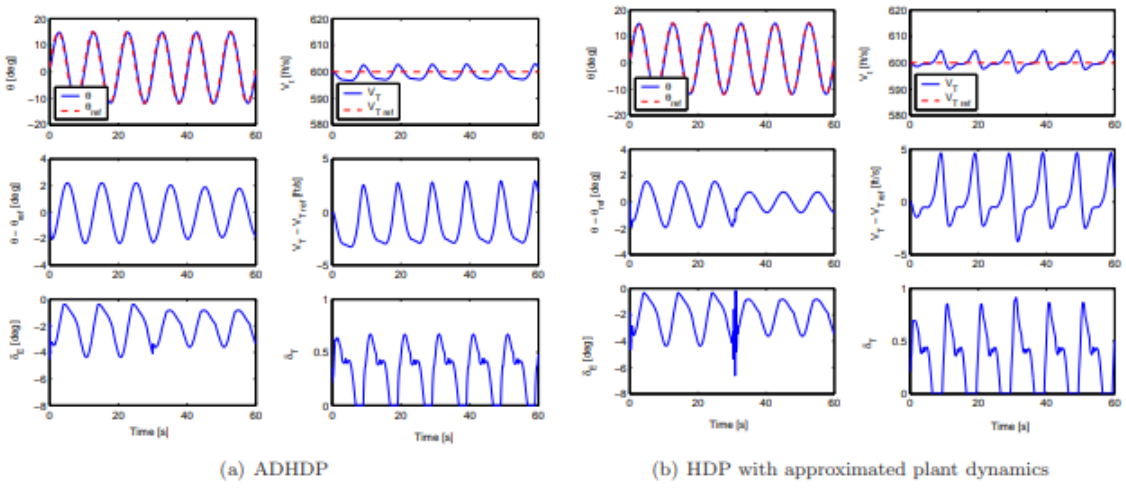
Some of the highlights in literature for adaptive control with RL for systems with input constraints are the work of Liu *et al.* (2015) who designed a novel RL-based robust adaptive control algorithm for optimal control of uncertain nonlinear systems which are subjected to input constraints to address the problem of real-time applications and offline implementations of model-free ADP algorithms. He showed that by designing a single critic NN for the approximation of the optimal control (HJB solutions), the uncertain continuous-time nonlinear system could converge to stability in less than a minute, without a need for initial stabilizing control. However, this approach also requires prior knowledge of the system dynamics, which is not always available.

3.5. Applications in Aerospace Systems

Reinforcement learning/approximate dynamic programming applications are not limited to a specific field of study. These methods have been used for different power systems control and even building air conditioning control (Si *et al.*, 2004). In the following, some of the applications of RL in the context of control of the aerospace vehicles are highlighted. Balakrishnan and Biega (1996) introduced the use of neural networks for adaptive critic designs in aircraft optimal control. His approach was used in controlling the angle of attack of an agile missile with constraints on the state variable. The optimization problem was defined for reversing the flight path angle of the missile completely in minimum time. It was shown that the NN-based controller is successful in achieving a near-optimal control for the missile steering in its Mach number flight envelope.

RL methods have been testified successfully in several literature available for helicopter autonomous flying using NDP methods by Enns and Si (2003), apprenticeship learning approaches by Abbeel *et al.* (2007) and the policy search approach with PEGASUS algorithm on a learned transition model by Kim *et al.* (2004). Andrew Ng research group used apprenticeship learning methods for learning to fly a helicopter using real flight test-data which then is used for model identification (Abbeel and Ng (2004), Abbeel *et al.* (2007). Enns and Si (2003) designed an Action Dependent Heuristic Dynamic Programming (ADHDP) controller for AH64-APache helicopter. In their study, the critic network is used to approximate the discounted reward-to-go-function. As is described in chapter 2 for ADHDP architecture, the standard back-propagation is used through the critic network to update the actor network.

van Kampen *et al.* (2006) focused on the model-free, self-learning and the adaptability of RL methods and their advantages in the field of re-configurable flight control, where accidental changes in the plant dynamics require control system adaptation. They describe the prominence of RL compared to supervised learning as being free from prior instructions for achieving a specific high-level goal and RL ability to come up with policies and actions without the restrictions of a structured model. Rather than instructing the controller with necessary control actions, which might not be available on every time instance, an RL controller is given the desired states to learn the optimal control actions by itself. By comparing two continuous actor/critic RL methods of Action Dependent Heuristic Dynamic Programming (ADHDP) and action independent Heuristic Dynamic Programming (HDP), it is concluded that the continuous domain brings a broader scope of applications for adaptive RL if compared to the discrete area.



(a) ADHDP (b) HDP with approximated plant dynamics

Figure (3.1) Comparing the robustness of ADHDP and HDP controllers in longitudinal control of F-16 when there is a sudden change in pitch moment coefficient (van Kampen *et al.*, 2006).

Ferrari *et al.* (2008) designed an adaptive NN controller using a constrained ADP approach and showed its online adaptability for a business jet and a Hawker Beechcraft Bonanza aircraft. This approach showed fast adaptability and high performance in the presence of effectors failure and variations in

aerodynamic parameters. In another study, ACD and NN are used for developing an adaptive controller for a business jet model. It is shown that the performance is kept with respect to offline specification in the occurrence of failures (Ferrari and Stengel, 2004).

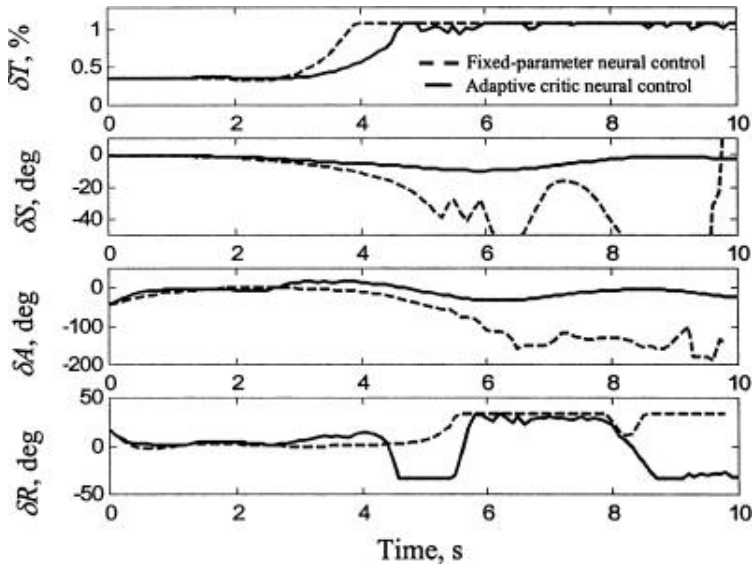


Figure (3.2) Comparison between the controller based on adaptive critic designs and the robust controller trained offline for $-70deg$ roll angle step command for fixed velocity of $160m/s$ and altitude of $7km$ (Ferrari *et al.*, 2008).

Naruta (2017) used CMAC function approximation in the NN framework for offline continuous action and state Q-learning in the task of flying a drone. Al-Tamimi *et al.* [123] developed an ADP-based model-free Q-learning scheme for discrete linear systems. His approach was tested in a simulation environment for a linearized F-16 aircraft model. In another study by Al-Tamimi *et al.* (2008), the proof of convergence is provided for the HDP scheme in solving the HJB equation for discrete-time nonlinear systems. Adaptive Critic/Actor-Critic structures with two NNS are used with an HDP algorithm for value iteration in solving the optimal control problem for the discrete-time nonlinear system. As van Kampen *et al.* (2006) described in his paper, the division of tasks between the actor and the critic in ACD architecture makes them attractive for researchers studying the design of the flight control systems.

Kiumarsi *et al.* (2016) used an off-policy approximated RL for online optimal tracking control and H control of an Unmanned Air Vehicle (UAV) system, without prior knowledge of the system dynamics. It is shown that the definition of utility function is critical in achieving a feasible solution to the optimal control problem. To deal with input constraints and assurance of feasibility a finite element method of L_2 condition is used. Kiumarsi *et al.* (2016) can be categorized in the same RL class that deals with safety, (Modares *et al.* (2014), Berkenkamp *et al.* (2017), Chow *et al.* (2018), Perkins and Barto (2002)).

Zhou *et al.* (2017) applied an incremental ADP (iADP) with full state feedback (iADP-FS) and with output feedback (iADP-OP) for stabilization of a non-linear second-order continuous missile model with aerodynamic uncertainties and rapid changes in the system. In Zhou (2018), iADP is applied to a non-linear Multiple-Input Multiple-Output (MIMO) attitude control problem of a spacecraft with liquid sloshing for two conditions of full and partial observable systems. The learning algorithms are developed for both off-line batch mode and online recursive adaptation. In Zhou *et al.* (2018), the near-optimal adaptive Incremental model-based Dual Heuristic Dynamic Programming (IDHP) for an online Adaptive Critic Design (ACD) is tested for the same missile model and a non-linear air vehicle model. This method identifies the incremental model online and eliminates the off-line learning while improving the control performance and convergence rate. Also, the method is validated for a fault-tolerant control task and with measurement noise.

de Vries 2017 experiment the application of discrete Q-learning on the Innovative Control Effector (ICE) aircraft. The results of the training with the discrete agent are given by Figure 3.3, and a more detailed description of the implementation is presented in chapter 4. Although the discrete approach

brought the study of the RL controller for ICE aircraft to a new level, however, it imposed some limitations that make the basis of this thesis study.

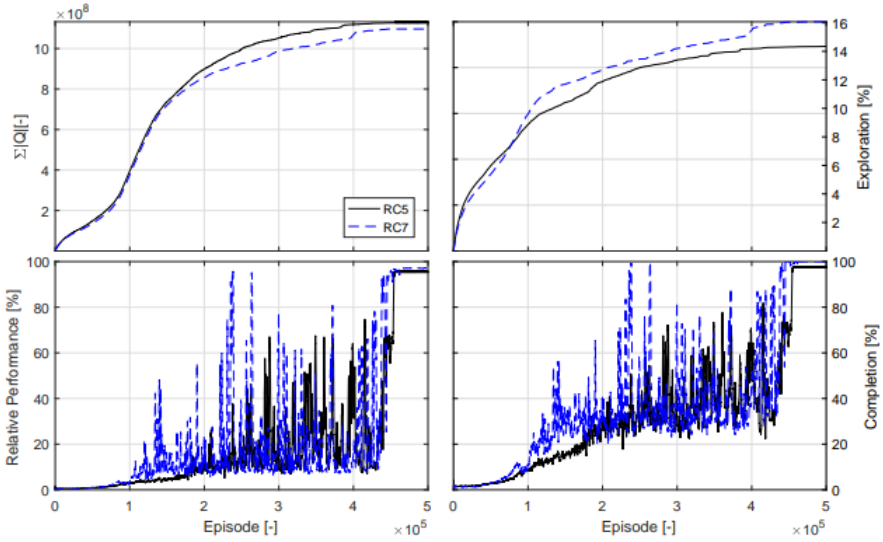


Figure (3.3) The Q-learning training results for longitudinal control of ICE aircraft showing the Q-values and learning performance for two nominal cases of bench marking (de Vries, 2017).

3.6. Conclusion

In this chapter, what reinforcement learning can offer in case of optimal and adaptive control was studied separately for each context, and its application for flight control system design was highlighted. It was shown that RL is related to optimal control by the introduction of cost-functions and error minimization (or maximization of value-function for Q-learning methods). For example, in using RL for optimal tracking control, the goal is to design an optimal control input to assure that the state of the system will follow the desired reference trajectory by minimizing a cost functional. To improve computational performance and memory usage, function approximators are used instead of tabulating RL components of the value function and policies in matrices. Tuning control parameters, however, still exist, and it becomes more significant for complex systems with a large number of states for which discrete methods are not applicable, and function approximation methods are required. Therefore, the successful application of many RL methods relies on a potent function approximator and its fine-tuning in terms of choosing the correct hyperparameters.

Optimization methods such gradient-descent were introduced, which are reliable methods for finding the local optima and are the most used methods for actor-critic structures. Also, it was elaborated how RL-based control can be seen as a model-free control strategy where the system learns a controller by itself (Grondman, 2015) either using batch flight data or while obtaining new data in online operation. Therefore, the nonlinear model of the system can directly be used by the controller while learning the nonlinear control law. Another main characteristic of RL control is being adaptive to changes in the system. This feature of RL was compared to other supervised adaptive controllers. In summary, what makes RL attractive for re-configurable flight controllers is its ability to adapt to the changes in the surrounding environment and the system dynamics. These characteristics of RL agents makes them a suitable candidate for self-learning or adapting flight controllers designed for a wide range of flight conditions and further for unforeseen circumstances.

In conclusion, RL has evolved as a promising technique for the design of self-learning, model-free, and adaptive controllers. One of the main advantages of this paradigm is that there is no condition imposed on the system from the RL controller and applies to all types of systems e.g., time-varying, stochastic, and nonlinear. Therefore, the design of an efficient controller does not need the desired

response as a prerequisite. The evolving of the controller is done in an online manner with frequent experience in its attempt for the control of the unknown system. However, recently, there are studies done to combine RL control with the Lyapunov theorem, which of course, impose some limitations for the RL controller but, on the other hand, ensure the stability of the system. This combination, however, is seen as hindering the exploration for RL agents and, therefore, is not welcomed by some of the researchers in the field. In the last section of this chapter, some of the leading successful applications of RL control in the aerospace field was presented. Most of these studies show the effectiveness of adaptive critic designs for FCS design of various aerospace vehicles from helicopters to small business jets and also innovative fighter aircraft.

4

Innovative Control Effector Aircraft (ICE) Model

With the introduction of new objectives for the design of the combat aircraft from the 1990s by the US Department of Defense, new standards for the airframe weights, maneuverability, and the overall cost of the aircraft were established (Bowlus *et al.*, 1997). At the same time, the practices with the configurations with low Radar Cross Section (RCS) proposed modifications to the external body of the aircraft. Tail-less aircraft configuration is one of the most promising solutions to new standards for weight reduction and low RCS. One of the most promising designs to meet these new objectives is the Innovative Control Effector (ICE) aircraft, which is a tail-less over-actuator aircraft designed by Lockheed Martin.

In this chapter, the control challenges of the ICE aircraft, the high-fidelity aerodynamic model, and the dynamics, including the equations of motion of the aircraft, are discussed. First, an introduction of the ICE program and its design objectives are given in section 4.1. Some of the primary purposes of the ICE aircraft design is mentioned by section 4.2. In section 4.3, the configuration, and control suite of the aircraft are discussed along with the control challenges. The challenges that arise for control of the ICE aircraft by its unique design are elaborated in section 4.4. The ICE aircraft aerodynamic model, which will be used for the central thesis, is described in section 4.5. The recent studies on application of reinforcement learning for ICE aircraft is presented in section 4.6. Finally, a conclusion is given in the chapter in section 4.7.

4.1. Introduction

ICE program established in 1993 in three main phases of conceptual design and analytic studies on an innovative control suite aircraft. The objective of the ICE model is to develop and validate innovative aerodynamic flight control effector suits for a tailless aircraft configuration (Bowlus *et al.*, 1997). Two concepts were suggested based on the application of the aircraft as land-based and carrier-based with the difference in structural elements and flying-wing configurations. Different combinations of control effectors and types was studied analytically for the two settings in the design phase. For the second phase, the wing designs were tested for collecting empirical data in the wind tunnel for developing an aerodynamic model of the aircraft and the effector as well as their interactions for different flight conditions (Dorsett and Mehl (1996), Matamoros Cid (2017). Figure 4.1 shows the configuration and control suites for the two designs. The land-based baseline configuration in Figure 4.1(a) or ICE 101-series is discussed in the following and will be referred to as ICE aircraft.

From a control perspective, most of the design objectives defined for the fighter aircraft are given as improved low signature characteristics with the traditional control surfaces; effectiveness for high angle of attack (AOA); tailless fighters applicability; decreasing weight and drag with conventional control suite; decreasing hinge moment and susceptibility to aeroelastic effects (Dorsett and Mehl,

1996). ICE model control poses a challenging test case due to its unstable nature, non-linear behavior, and state-space dimensionality. In the following the aircraft configuration and the control challenges associated with it are illustrated. Next, a detail description on the models available are given.

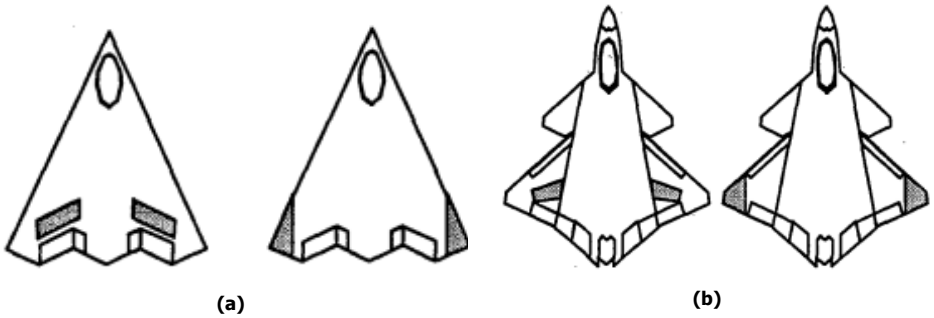


Figure (4.1) Control suits in analytic study for (a) land-based and (b) carrier-based configurations (Dorsett and Mehl, 1996)

4.2. ICE Aircraft Control Objectives

The ICE aircraft designed control objectives are mainly based upon maneuverability, stealth, and resilience. Maneuverability, also known as agility, is essential to gain tactical advantages in close-quarter air-to-air combat and to evade enemy surface-to-air missiles (SAM) when other counter-measures have failed. The ICE aircraft configuration is designed with Relaxed Static Stability (RSS) in both pitch and yaw axes to achieve high maneuverability. From 1974 and by the first flight of the General Dynamics (now Lockheed Martin) F-16 Fighting Falcon, the next generation of highly maneuverable fighters introduced that take advantage of the inherently aerodynamic instability to obtain extraordinary agility while relying on FCC to stabilize the aircraft for the human pilot. Most 4th and 5th generations fighter combat aircraft, like the Lockheed Martin F-22 and F-35, the Eurofighter and the Sukhoi Su-57, follow the same concept for inherent instability. Flying an aircraft with RSS is achieved by fitting stability augmentation devices and with the constant intervention of the FCS to maintain the aircraft level. Therefore, the continuous intervention of an automated FSC is also critical and vital for ICE design.

Stealth features for a combat aircraft highly connect with survivability on the modern battlefield and effectiveness in strike missions. This allows to penetrate enemy defenses, achieve surprise and keep the initiative in air-to-air combat, in particular at long range. The three main fields of research in this direction are reductions of radar signature, infra-red emissions, and active electromagnetic emissions. The first aspect is achievable by reducing the radar cross-section (RCS), e.g., avoiding surfaces with right angles, aligning aircraft edges in few particular directions, and using S-shaped intake ducts to hide the engine front face (Whitford, 1987). Further reductions in RCS can be achieved with the elimination of vertical surfaces (Dorsett and Mehl, 1996), a technique used operationally only in the Boeing B-2 Spirit, a stealth heavy bomber with very little maneuverability. ICE aircraft is, therefore, designed without horizontal and vertical tails. In a tail-less aircraft, other advantages can be obtained apart from low RCS, as the flying-wing configuration is very efficient in producing lift, allowing for overall savings in weight and cost. Without the tail, the stabilizing moments in pitch and yaw for ICE have to be generated with the proper actuation of the redundant control surfaces, a challenge clearly defined in the context of the ICE development objectives (Bowlus *et al.*, 1997).

In general, there are much higher chances for military aircraft to receive damage during their operational life compared to civilian ones (Van Kampen *et al.*, 2006). Therefore, resilience is critical for the redundant ICE model that uses 13 control effectors. For an unmanned aircraft, resilience is defined as the ability to sustain battle, even with failures in the system. For a manned aircraft, the pilot can intervene in making high-level decisions when confronted with unforeseen situations, as the need for diversions or the adoption of defensive counter-measures. With less involvement of the human pilots, most of the capabilities such as planning and decision making have to be performed by the control system. Therefore, achieving the same level of resilience as a manned aircraft will be much challeng-

ing for a UAV. Improvements in automatic adaptive systems can tackle this incentive problem. In the occurrence of accidents such as signal loss and other failures, adaptability and the ability to control the aircraft remain completely to the smart FCC on-board.

4.3. ICE Aircraft Control Suites

The land-based baseline configuration of the ICE simulation model is described by Buffington and Buffington (1997), like a single supersonic engine, 65 deg sweep delta flying wing and a multi-role fighter. The ICE aircraft has highly redundant 13 multi-axis control effectors (Buffington and Buffington, 1997). A schematic of the control suits for ICE is shown in Figure 4.4. The control effectors include Leading edge flaps (LEF), all moving wing tips (AMT) and spoiler-slot deflectors (SSD) which are mainly used for lateral-directional control, the pitch-flaps (PF) to provide longitudinal control power and elevons and multi-axis thrust vectoring (MTV) that are used for both symmetric and asymmetric cases. These control effector are described in more detail below, inspired by the work of Dorsett and Mehl (1996), Matamoros Cid (2017) and de Vries (2017).

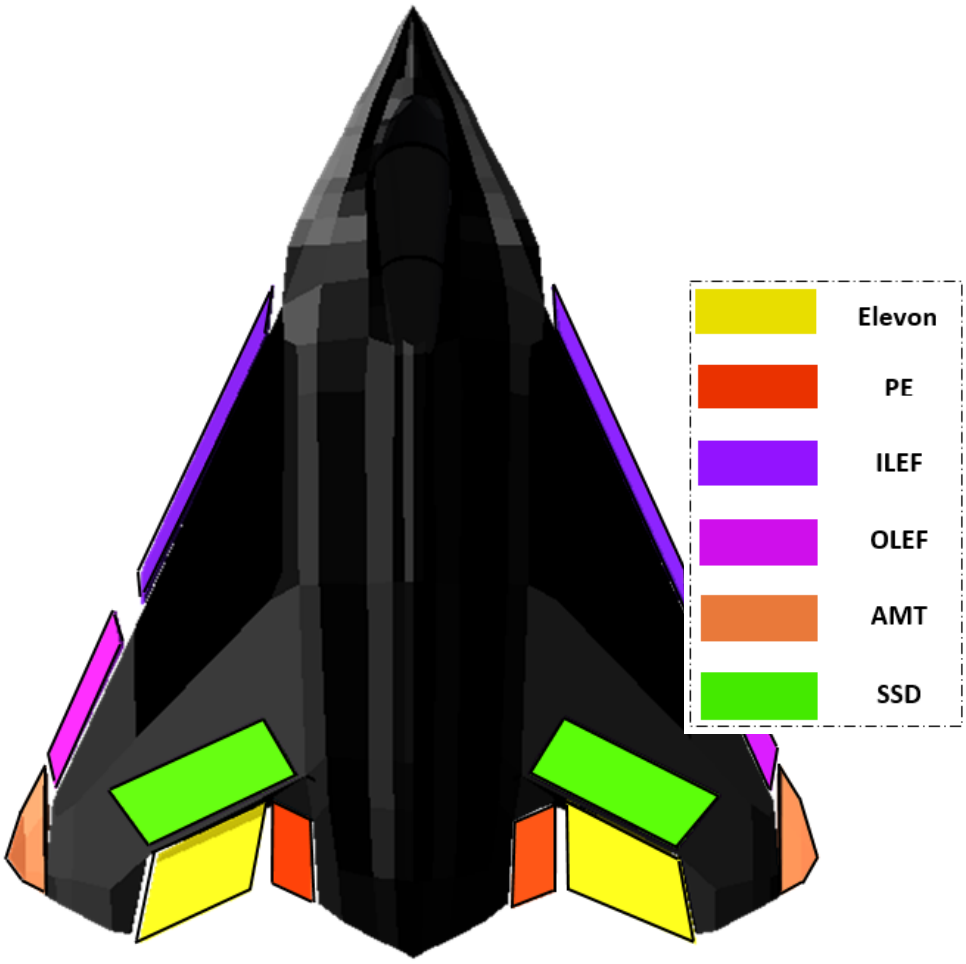


Figure (4.2) ICE aircraft control suite

Elevons

Two elevons (or middle trailing edge flaps) that can deflect independently and asymmetrically for roll control and also contribute to pitch control by symmetric deflections between -30 and 30 degrees. The control power of elevon deflections is affected mainly by the SSDs. The effect of elevons on yaw

characteristics that can cause by the parasite and induced drags are little.

Pitch flaps (PF)

The same as elevons, pitch flaps can deflect both symmetrically and asymmetrically while they are dedicated to the pitch attitude control of the aircraft. In particular, in the longitudinal trimming of the aircraft in high angle of attacks and when the control power of other effectors is deteriorated. PEs deflect indifferently between -30 and 30 degrees limit.

Inboard leading edge flaps (ILEF)

A pair of ILEFs use for lateral-directional control in high angle of attacks (AOAs) and low speeds. These effectors increase the drag with their downward deflection up to 40 degrees (tested for 0, 20, and 40 degrees) and causing adverse yaw and therefore stabilizing the aircraft in its roll coordination. The control effectiveness for these effectors are nonlinear and a function of AOA with high interaction with outboard LEFs (OLEFs).

Outboard leading edge flaps (OLEF)

Another pair of leading edge flaps with the same deflection limits as ILEFs. The interaction of these two pairs of effector avoids flow separation from the wing, particularly for high angle of attacks. The integration of LEFs to match up with straight or deflected AMT was carefully studied to facilitate the control effectiveness of other control effectors effectively.

All moving tips (AMT)

The deflection of the trailing edges in two directions of up (TEU) and down (TED) are designed as yaw control devices by producing axial forces and yaw moments. The unique aft-swept hinge-line contributes to the yaw control by providing considerable side-forces. The skewed hinge-line also adds to axial forces and produces roll control power for a different angle of attacks, e.g., for AOA of below 10 degrees; an adverse roll can be generated by TED. AMTS are the primary sources for yaw control in high angle of attacks and can maintain their effectiveness for wide ranges in the speed envelope of the aircraft. The differential AMTs can deflect between -10 to 60 degrees.

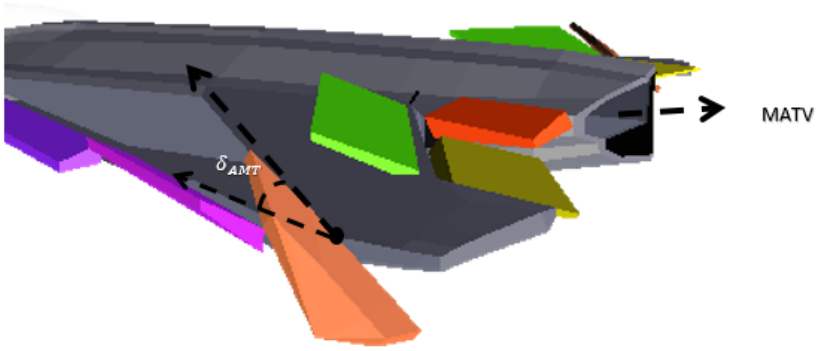


Figure (4.3) AMTs deflection in perspective view of the ICE model

Spoiler slot deflectors (SSD)

While AMT is used for generating yaw moments in the higher angle of attacks (e.g., more than 20 degrees), the spoiler slot deflectors (SSD) is more effective for yaw augmentations in lower angle of attacks (for the higher angle of attacks the intervention of the thrust vectoring is required). SSDs can be used for spin recovery or anti-spin control due to their sufficient control powers in the flight envelope of the aircraft. The control surfaces aft of the SSDs can become curtailed due to the interaction with the SSDs. Therefore, SSDs are located in outboard and forward locations on both upper and lower surfaces of the aircraft with different hinge points and consequently able to deflect in opposite directions (e.g., when the top spoiler deflect upwards the other spoiler can deflect downward) so that the flow will be

led within the surface with least footprint of the trailing edge effectors. SSDs can deflect up to 60 degrees.

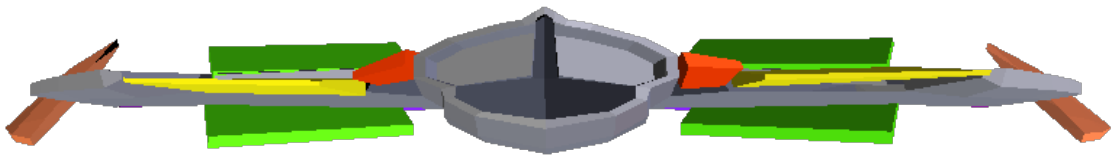


Figure (4.4) SSDs deflection in back view of ICE model

Multi axis thrust vectoring (MATV)

Integrating thrust vectoring power to the control power substantially improves the potential of the aircraft for unlimited maneuverability. MATV is practiced for F-16, Sukhoi 27 and 35 and F-22 Raptor and showed tremendous benefits for post-stall and high AOA low-speed close-in-air combats. MATV is designed for both pitch and yaw thrust vectoring for ICE aircraft. In addition, it enables for the ICE configuration an unlimited AOA potentials when used in combination with aerodynamics effectors. The recent designs for ICE consider fluidic thrust vectoring instead of the conventional nozzle-based thrust vectoring. The MATV is tested for a maximum of 15 degrees vector angle with the possibility of omnidirectional deflection. MATV has more power in lower aircraft speeds, and for higher speeds, the MATV becomes less effective, and the aircraft control becomes dependent on the aerodynamic surfaces. MATV also allow the full utilization of the roll control power by integration with SSDs and by coordinating the yaw control.

In the table below, the list of the control suites as well as their position and rate limits are described.

Table (4.1) ICE aircraft control suite specification

Effector	Position limits [deg]	Rate limits [deg]
Elevon	[-30, 30]	150
PF	[-30, 30]	150
ILEF	[0, 40]	40
OLEF	[-40, 40]	40
AMT	[0,60]	150
SSD	[0, 60]	150
MATV	[-15,15]	150

So far, the nonlinear actuator models which consider the hinge moment coefficients are not yet integrated to the model. However, it is recommended that for leading-edge actuators dynamics (i.e., inboard and outboard LEFs) low-bandwidth transfer functions be used while for the rest of the actuators, including thrust vectoring, the high-bandwidth transfer functions are suggested, as given by Equation 4.1.

$$H_l(s) = \frac{18times100}{(s+18)\times(s+100)}, \qquad H_h(s) = \frac{40times100}{(s+40)\times(s+100)} \tag{4.1}$$

The control power defined by the control moments generated by each effector at its extreme deflections is depicted in Figure 4.5. It can be seen that all the effectors influence the three pitch, roll, and yaw controls (it is concluded statistically that these influences are also significant Niestroy *et al.* (2017)). The detailed description on the interactions between the effectors can be found in Gillard (1998).

Figure (4.5) The effect of left side control deflections on their maximum deflection for $M = 0.3$ in different AOA on body axis (a) pitch, (b) roll and (c) yaw controls Niestroy *et al.* (2017).

4.4. ICE Aircraft Control Problem and Challenges

The control problem for ICE aircraft can be tackled in two ways: the optimal CA problem and the optimal tracking problem. Considering the tailless configuration of the ICE aircraft and its redundancy in the number of control effectors, several control challenges arise. These include the longitudinal and specifically lateral-directional control, which is inherited in the tailless configuration, the high nonlinearity in the aircraft model, and the effectors' utilization. Moreover, the control allocation problem, which includes a suitable distribution of control power between available effectors in achieving a control task, is another challenge induced by ICE's unique design. These control and stability challenges of the ICE aircraft have been addressed in Bowlus *et al.* (1997), Eberhardt and Ward (1999), Ngo *et al.* (1996), Matamoros Cid (2017) and de Vries (2017). The stability, performance, and plant uncertainties are addressed in Ngo *et al.* (1996) as part of a manual on the flight control systems of the aircraft. The stability and control challenges of tailless configurations are discussed in Bowlus *et al.* (1997). Another

documentation on basic control allocation for the ICE aircraft is presented in Buffington (1999). An indirect adaptive flight control system is proposed in Eberhardt and Ward (1999). The control allocation is another main challenge for ICE aircraft with redundancy in multi-axis control effectors. Also, it is shown in Matamoros Cid (2017) that ICE actuators have highly nonlinear control effectiveness and are coupled and interactive. A study on ICE aircraft effectors control effectiveness is given by Addington and Myatt (2000).

Some of the most recent studies by Matamoros Cid (2017) include model-based control allocation for lateral-directional of the aircraft in real-time. The control allocation challenges are defined in this study as the nonlinearity of the actuators control effectiveness and the nonlinear interaction of the control effectors. The conventional CA methods with the assumption of linearity between actuator positions and control-induced forces and moments fail at providing adequate performance for more complex and innovative tail-less aircraft designs with directional control powers. In a recent study at the Delft University of Technology, an INCA-based FCS was developed for the nonlinear CA of the ICE model in real-time. This approach produced promising results when compared to Linear CA (LCA) methods, for tracking and CA performance of the aircraft in lateral-directional control and aggressive maneuvers. The limitation with this approach is the assumptions of full state feedback in the control loop and it being a robust model-based method.

In another study by de Vries (2017), which testifies the reinforcement learning-based CA methods and Q-learning for longitudinal control of the ICE aircraft in discrete state and action spaces. In this study, an offline discrete and tabular RL-based control allocation method (Q-learning) was developed for the ICE model to fulfill an abstract objective of altitude reference tracking control. It was found that although RL is capable of achieving the aim of retaining a reference altitude, being initialized with an off-set, the performance of the discrete RL is less than that of model-based control methods (such as INCA). This shortcoming is expected to be due to the limitations in the function approximations. The same study proposes, for further studying, a hybrid approach of combining model-bounded CA with RL-based CA methods, with the former to be used in normal flight conditions and the latter in unforeseen circumstances.

This thesis, however, addresses two problems of longitudinal control and control allocation of the ICE aircraft using reinforcement learning control in continuous state and action spaces.

4.5. ICE Aircraft Dynamics and Aerodynamic Model

The ICE simulation model is available by the framework MATLAB/SIMULINK, written by version R2012b. Lockheed Martin has released wind tunnel test data as a high-fidelity aerodynamic database for the ICE aircraft. The aerodynamic coefficients for the ICE model are defined in body reference frame $\mathcal{F}^b$ as the right-hand axis system with the origin at the aircraft center of gravity and the x-axis in the direction of the nose and y-axis pointing to the right and the z-axis pointing downwards. The lookup tables coefficients are defined in the aerodynamic model with reference to body frame and 38% mean aerodynamic chord (MAC). The right-hand axis system has its origin at the aircraft center of gravity and the x-axis pointing toward the back of the aircraft and the z-axis pointing upwards and the y-axis to the right. In the following, first, this model is described in subsection 4.5.1 as the main and basis of the aerodynamic models available for the aircraft. Another spline-based model recently released by () is a then discussed in subsection 4.5.2.

4.5.1. Six degrees of freedom aerodynamic model

The Aerodynamic Model (AM) released by Lockheed Martin Aeronautics (LM Aero) for academic use is developed in six degrees of freedom (6-DoF) from experimental wind tunnel data obtained by more than 900 hours of testing resulted in 108 lookup tables. A detailed description of the model and trim maps can be found in Niestroy *et al.* (2017). The data includes the AMT and SSD Mach effects, the derivatives of dynamics, and the hinge moments derivatives, in addition to the control effector interactions and the aeroelastic effects. The model, however, does not include the engine model, since statistical anal-

ysis is performed on it. The aerodynamic force and moment coefficients are then given based on the lookup tables nonlinear variables. The aerodynamic forces $[F_x F_y F_z]^T$ and moments $[M_x M_y M_z]^T$ which are defined in body frame $\mathcal{F}^b$ can be derived from dimensionless and normalized force and moment coefficients as:

$$C_{F_x} = \frac{F_x}{\frac{1}{2}\rho V^2 S}, \quad C_{F_y} = \frac{F_y}{\frac{1}{2}\rho V^2 S}, \quad C_{F_z} = \frac{F_z}{\frac{1}{2}\rho V^2 S} \tag{4.2}$$

$$C_{M_x} = \frac{M_x}{\frac{1}{2}\rho V^2 S b}, \quad C_{M_y} = \frac{M_y}{\frac{1}{2}\rho V^2 S \bar{c}}, \quad C_{M_z} = \frac{M_z}{\frac{1}{2}\rho V^2 S b} \tag{4.3}$$

where $\rho(slug/ft^3)$ is the air density, $V(ft/sec)$ is the true airspeed, $S(ft^2)$ is the wing area, $b(ft)$ is the span of the wing and $\bar{c}(ft)$ is the mean aerodynamic chord. Each aerodynamic coefficient, by itself, is defined by the lookup tables nonlinear terms summation with each term being a function of the control effector positions and the aircraft operating points $(\alpha, \beta, \delta, M)$, where α is the angle of attack, β is the sideslip angle, M is the Mach number and δ is the effectors position. In Table 4.2, the range of this operation points are given.

Table (4.2) 6-DoF aerodynamic model operating points range

Operating point	Operation domain
α [deg]	[-4, 90]
β [deg]	[-30, 30]
M [-]	[0.1, 2.16]

From the model, the aerodynamic coefficients are derived by interpolation and linear combination of the lookup tables coefficients. The lookup tables coefficients, however, are a nonlinear function of the operating points and other lookup tables coefficients. The coefficients and their dependency are given in Table 4.3.

Table (4.3) 6-DoF aerodynamic model operating points range

Tabular Coefficients	Dependency	Tabular Coefficients	Dependency
$C_{F_{x_1}}, C_{M_{x_1}}$	α, M	$C_{F_{x_{10}}}, C_{M_{x_{10}}}$	$\alpha, \delta_{OLEF_R}, \delta_{AMT_R}$
$C_{F_{x_2}}, C_{M_{x_2}}$	α, β, M	$C_{F_{x_{11}}}, C_{M_{x_{11}}}$	$\alpha, \delta_{AMT_L}, \delta_{E_L}$
$C_{F_{x_3}}, C_{M_{x_3}}$	$\alpha, M, \delta_{E_L}, \delta_{SSD_L}$	$C_{F_{x_{12}}}, C_{M_{x_{12}}}$	$\alpha, \delta_{AMT_R}, \delta_{E_R}$
$C_{F_{x_4}}, C_{M_{x_4}}$	$\alpha, M, \delta_{E_R}, \delta_{SSD_R}$	$C_{F_{x_{13}}}, C_{M_{x_{13}}}$	$\alpha, M, \delta_{PF}, \delta_{SSD_L}, \delta_{SSD_R}$
$C_{F_{x_5}}, C_{M_{x_5}}$	$\alpha, \beta, \delta_{ILEF_L}$	$C_{F_{x_{14}}}, C_{M_{x_{14}}}$	$\alpha, \beta, \delta_{AMT_L}$
$C_{F_{x_6}}, C_{M_{x_6}}$	$\alpha, \beta, \delta_{ILEF_R}$	$C_{F_{x_{15}}}, C_{M_{x_{15}}}$	$\alpha, \beta, \delta_{AMT_R}$
$C_{F_{x_7}}, C_{M_{x_7}}$	$\alpha, \beta, M, \delta_{ILEF_L}, \delta_{OLEF_L}$	$C_{F_{x_{16}}}, C_{M_{x_{16}}}$	$\alpha, \beta, \delta_{SSD_L}$
$C_{F_{x_8}}, C_{M_{x_8}}$	$\alpha, \beta, M, \delta_{ILEF_R}, \delta_{OLEF_R}$	$C_{F_{x_{17}}}, C_{M_{x_{17}}}$	$\alpha, \beta, \delta_{SSD_R}$
$C_{F_{x_9}}, C_{M_{x_9}}$	$\alpha, \delta_{OLEF_L}, \delta_{AMT_L}$	$C_{F_{x_{18}}}, C_{M_{x_{18}}}$	α, M

In Equation 4.4 - 4.9 it is shown the aerodynamic coefficients construction based on the tabular coefficients.

$$C_{F_x} = \sum_{n=1}^{17} C_{F_{x_n}} \tag{4.4}$$

$$\begin{aligned} C_{F_y} =& C_{F_{y_1}} + C_{F_{y_2}} + C_{F_{y_3}} + C_{F_{y_6}} + C_{F_{y_8}} + C_{F_{y_9}} + C_{F_{y_{11}}} + C_{F_{y_{13}}} + C_{F_{y_{14}}} + C_{F_{y_{16}}} \\ & - (C_{F_{y_4}} + C_{F_{y_5}} + C_{F_{y_7}} + C_{F_{y_{10}}} + C_{F_{y_{12}}} + C_{F_{y_{15}}} + C_{F_{y_{17}}}) \end{aligned} \tag{4.5}$$

$$C_{F_z} = \sum_{n=1}^{17} C_{F_{z_n}} + \frac{\bar{c}q}{2V} C_{F_{z_{18}}} \tag{4.6}$$

$$\begin{aligned} C_{M_x} =& C_{M_{x_1}} + C_{M_{x_2}} + C_{M_{x_3}} + C_{M_{x_6}} + C_{M_{x_8}} + C_{M_{x_9}} + C_{M_{x_{11}}} + C_{M_{x_{13}}} + C_{M_{x_{14}}} + C_{M_{x_{16}}} \\ & - (C_{M_{x_4}} + C_{M_{x_5}} + C_{M_{x_7}} + C_{M_{x_{10}}} + C_{M_{x_{12}}} + C_{M_{x_{15}}} + C_{M_{x_{17}}}) + \frac{bp}{2V} C_{M_{x_{18}}} + \frac{br}{2V} C_{M_{x_{19}}} \end{aligned} \tag{4.7}$$

$$C_{M_y} = \sum_{n=1}^{17} C_{M_{y_n}} + \frac{\bar{c}q}{2V} C_{M_{z_{18}}} \tag{4.8}$$

$$\begin{aligned} C_{M_z} =& C_{M_{z_1}} + C_{M_{z_2}} + C_{M_{z_3}} + C_{M_{z_6}} + C_{M_{z_8}} + C_{M_{z_9}} + C_{M_{z_{11}}} + C_{M_{z_{13}}} + C_{M_{z_{14}}} + C_{M_{z_{16}}} \\ & - (C_{M_{z_4}} + C_{M_{z_5}} + C_{M_{z_7}} + C_{M_{z_{10}}} + C_{M_{z_{12}}} + C_{M_{z_{15}}} + C_{M_{z_{17}}}) + \frac{bp}{2V} C_{M_{z_{18}}} + \frac{br}{2V} C_{M_{z_{19}}} \end{aligned} \tag{4.9}$$

Although there are similarities between the coefficients, however, they have different values for various forces and moments and need to be estimated. The control input vector δ includes the 11 aerodynamic control surfaces and two thrust vectoring inputs for yaw and pitch in addition to the throttle power. Therefore, 13 effectors are included by the control vector in total. The action zone of these effectors is limited by the constraints imposed by the saturation limits of the actuators. However, the multivariate thrust vectoring deflection is limited by circular constraints as formulated by Matamoros Cid (2017). In the same study, the stability analysis of the aircraft can be found. Since the 6-DoF model is not analytical and uses tabulated data the transition matrices, e.g., $\partial x(t+1)\ \partial u(t)$ and $\partial x(t+1)\ \partial x(t)$ cannot be obtained by analytically and numerical methods should be used.

4.5.2. Three degrees of freedom aerodynamic model

Apart from the original simulink-based ICE model, a spline-based model was developed by van der Peijl (2017) for calculating the force and moments coefficients in MATLAB environment. This model was used by de Vries (2017) in his study on design of a discrete RL-based controller for ICE aircraft. In the spline model the aerodynamic coefficients are identified with a spline approximation as a function of the $(\alpha, M, \beta, V, p, q, r, T, u)$, where $V(ft/s)$ is the true air speed, p, q and $r(rad/s)$ are the body rotation rates in x, y and z axes, $T(lbf)$ is the thrust and u is the input vector given by:

$$\begin{aligned} \mathbf{u} =& [\delta_{ILEF_L}, \delta_{OLEF_L}, \delta_{AMT_L}, \delta_{E_L}, \delta_{SSD_L}, \delta_{PF}, \\ & \delta_{ILEF_R}, \delta_{OLEF_R}, \delta_{AMT_R}, \delta_{E_R}, \delta_{SSD_R}, \delta_{pTV}, \delta_{yTV}] \end{aligned} \tag{4.10}$$

where L and R denotes to the left and right effector and pTV and yTV corresponds to the thrust vectoring for pitch and yaw movements. Including the thrust vector projection on the body frame $\mathcal{F}^b$ in the derivation of the aerodynamic forces and moments, the aerodynamic coefficients are derived by:

Table (4.4) Force and moment coefficients derived by spline-based ICE model

Force Coefficients		Moments Coefficients	
Definition	**Equation**	**Definition**	**Equation**
C_{F_x}	$\frac{-F_x + T cos(\delta_{pTV}) cos(\delta_y TV)}{\frac{1}{2}\rho V^2 S}$	C_{M_x}	$\frac{M_x}{\frac{1}{2}\rho V^2 S b}$
C_{F_y}	$\frac{F_y - T cos(\delta_{pTV}) sin(\delta_y TV)}{\frac{1}{2}\rho V^2 S}$	C_{M_y}	$\frac{M_y + T d_{TV} sin(\delta_{pTV}) cos(\delta_{yTV})}{\frac{1}{2}\rho V^2 S \bar{c}}$
C_{F_z}	$\frac{-F_z - T sin(\delta_{pTV}) cos(\delta_y TV)}{\frac{1}{2}\rho V^2 S}$	C_{M_z}	$\frac{M_z + T d_{TV} cos(\delta_{pTV}) sin(\delta_{yTV})}{\frac{1}{2}\rho V^2 S \bar{c}}$

where $d_{TV}[ft]$ is the thrust vectoring arm from aircraft center of gravity and the forces and moments are in reference body frame $\mathcal{F}^b$ in $[lbf]$ and $[lbf.ft]$, respectively. The International Standard Atmosphere (ISA) is used for describing the atmospheric changes and conditions.

In the spline ICE model, the interpolation extrapolation is a combination of linear extrapolation and cubic spline interpolation. The reason is to generate more data points to improve the quality of the regression model. However, due to the high non-linearity of the model, higher degree splines are needed to result in an accurate model of the aircraft, which increases the computational cost. The study by van der Peijl (2017) concluded that zero-order of continuity would "likely" be enough for control allocation, and higher order of continuity is not suggested due to a reduction in the accuracy. Therefore, a trade-off between computational cost and accuracy is needed.

4.6. State-of-the-art in ICE Aircraft Control Solutions

de Vries (2017) used RL for solving the control allocation problem in achieving a control regulation problem. In his study, an off-line discrete RL controller is designed based on Q-learning for altitude control of the ICE model. The purpose of his research was to find whether and how an RL method can solve a CA problem. Since the tuning of the function approximators is often time-consuming, simpler approaches (discrete, tabulated) to RL is considered in his paper. He mentioned as one of the next steps to implement function approximation for performance improvement. His approach is particularly studied in chapter 4. Figure 4.6 shows the result of the experiment for discrete Q-learning control allocation, when the aircraft model is initialized with an offset of $500ft$ altitude and the regulation problem is to stabilize the aircraft in the desired altitude. Oscillations around the desired trajectory, as well as high variation in thrust and state outputs, can be seen from the plots. Also, the control policy that the controller obtained shows an un-smooth response.

Table (4.5) Conditions for two benchmarking cases of 5 and 7

Case	Effectors	Reward function
$RC5$	Elevon, PF, ILEF, OLEF	$\hat{h}$
$RC7$	Elevon, PF, ILEF, OLEF	$\hat{h}, \hat{\mathbf{u}}, \hat{T}$

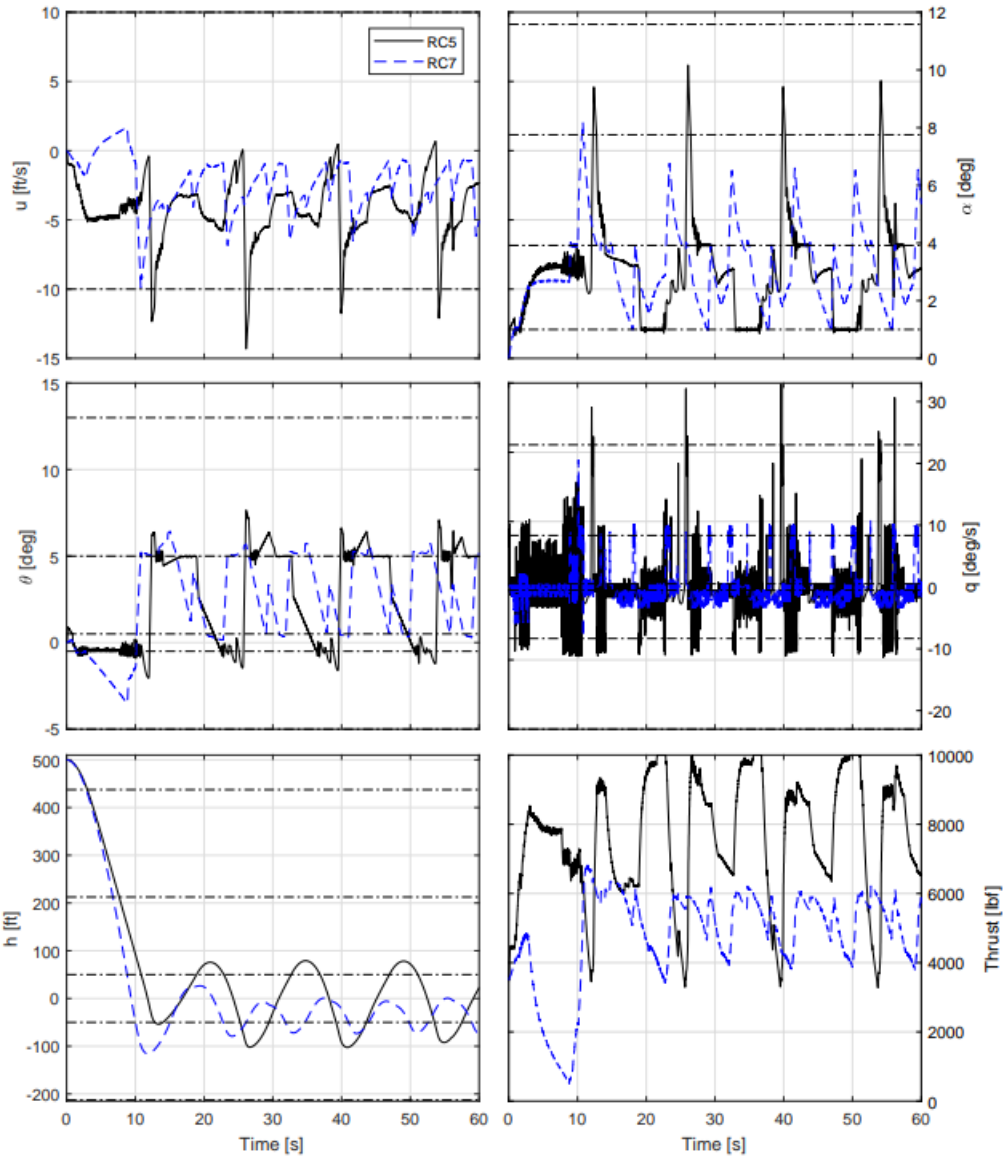


Figure (4.6) Q-learning control allocation results for ICE aircraft longitudinal control with objective of altitude tracking showing the body velocity $u[ft/s]$, the angle of attack $\alpha[deg]$, the pitch angle $\theta[deg]$, the pitch rate $q[deg/s]$, the altitude $h[ft]$ and the thrust value in $[lbf]$ for cases of 5 and 7 (de Vries, 2017).

Figure 4.7 shows the history of the inputs given to the agents for both of the benchmarks described in Table 4.5. It can be seen that while the control effort for case 5 is higher for case 7, as the OLEF the agent in case 5 has not used the OLEF, the thrust used for case 7 is also smaller, and the resulting tracking has less oscillation amplitude. Therefore, adding more constraints on the controller (in this case for the control effort and use of thrust) as imposed by the reward function can result in better performance by the controller, as can be seen by results given in Table 4.6. The general conclusion from this study was that discrete RL shows limitations when it comes to a complex system like ICE aircraft. It is expected that approximate RL can overcome some of these deficiencies.

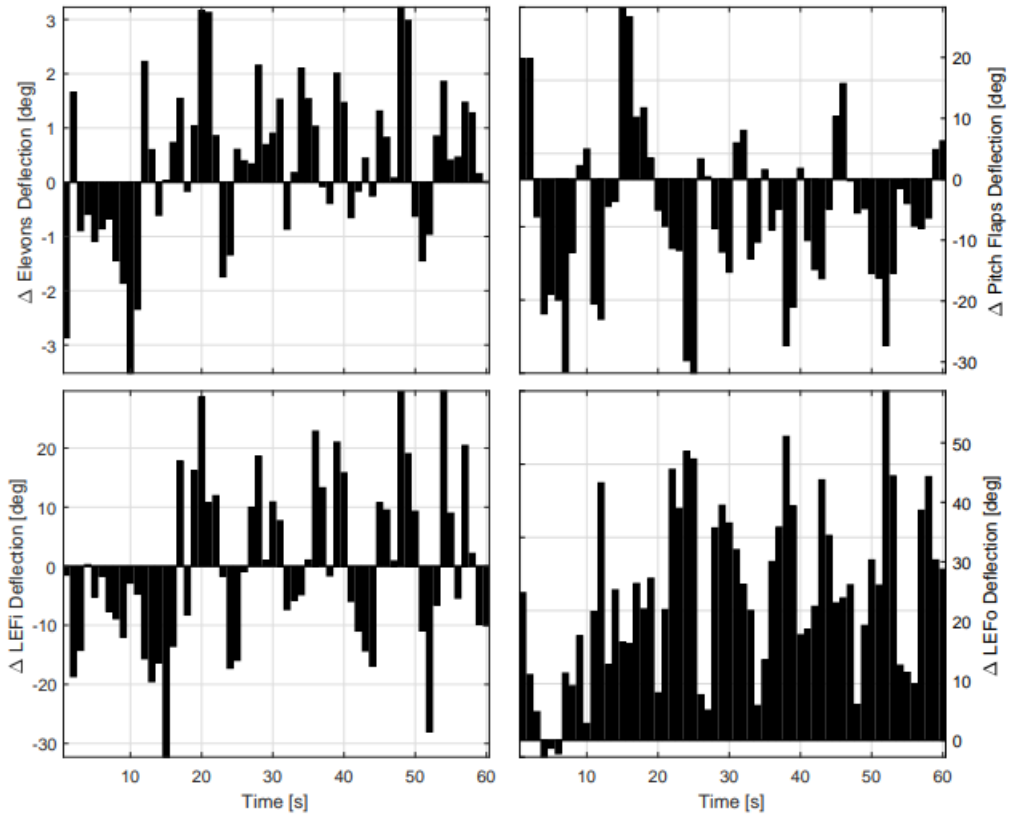


Figure (4.7) The inputs to the Q-learning control allocation algorithm for ICE aircraft longitudinal control with objective of altitude tracking showing the elevon deflection in $[deg]$, the PF deflection in $[deg]$, the ILEF deflection in $[deg]$, the OLEF deflection in $[deg]$, for cases of 5 and 7 (de Vries, 2017).

Table (4.6) Benchmarking results for cases of 5 and 7 (de Vries, 2017)

$RMS(h)$ $[ft]$	CI [%]	$RMS(\delta_E)$ $[deg]$	$RMS(\delta_{PF})$ $[deg]$	$RMS(\delta_{ILEF})$ $[deg]$	$RMS(\delta_{OLEF})$ $[deg]$	$RMS(T)$ $[lbf]$
154.9	55.84	18.23	22.47	26.03	29.13	7791
145.3	33.59	8.759	11.36	26.68	21.30	5008

4.7. Conclusion

As discussed by this chapter, the ICE aircraft design is not conventional and differs in its tailless configuration and more significantly in its redundancy in a number of control effectors and their assignment for control tasks from traditional combat aircraft. The ICE aircraft's main objectives for maneuverability, stealth, and resilience, which resulted in its unique design, propose some control challenges for its longitudinal and lateral-directional control mostly imposed by the tailless configuration. The over-actuated dynamics is suggested to contribute to the control ability by adding control power to the three axes of the aircraft. The control suite of ICE aircraft consists of 13 effectors that are used in the nonlinear model construction. So far, two models are developed for the aircraft. The 6-DoF model developed by Lockheed Martin in MATLAB/SIMULINK environment, which uses a combination of highly nonlinear 108 models (look-up tables) for force and moment coefficient estimations. The 3-DoF parametric spline model is developed in MATLAB framework, which creates an efficient way of storing the aircraft model and is also suitable for online adaptation with improving in adaptation if nullspace projection

for regression is stored and used during the online model learning. For more detailed information on the history of the ICE program and more detailed discussions on the control allocation, control suite configuration and the aerodynamic models the reader is referred to Dorsett and Mehl (1996), Gillard *et al.* (2006), Niestroy *et al.* (2017) and van der Peijl (2017).

5

Preliminery Implementation and Results

The main goal of this thesis is to apply a proper RL algorithm for continuous state and action spaces for the ICE aircraft. ICE aircraft case study is a complicated problem due to its nonlinear behavior and over-actuation, which make an under-determined system. However, in previous sections, it is shown how continuous reinforcement learning approaches, especially the actor-critic structures are suitable for handling the nonlinearities due to their model-free nature and ability to solve the control allocation problem while achieving the optimal control objective. However, to motivate the application of the continuous RL for the ICE aircraft and its outperformance to discrete RL, a more straightforward problem of pendulum swing-up would be discussed in the preliminary phase. This control problem is usually used as a benchmark test base for different control algorithms. In this chapter, after the introduction in section 5.1, the results for two different case studies are presented by section 5.2 and section 5.3. A conclusion based on the preliminery results are given in section 5.4.

5.1. Introduction

Pendulum Swing-up (PS) is a well-known control problem to control researchers due to the control challenges it provokes caused by its inherent instability and nonlinear behavior. This problem and a simpler but similar problem of pendulum on a moving cart has been discussed in reinforcement learning literature by Anderson (1989), Sutton *et al.*) (1992), Williams (1992) and Doya (2000). Although much of the developments in RL literature focused on discrete implementations, recently continuous/approximate methods have also been studied by researchers in the field to solve the problem of dimensionality and provide more generalization for real-world applications.

In this chapter, a discrete framework of Q-learning is implemented on a nonlinear control problem of a simpler case study of an inverted pendulum, described by subsection 5.2.1. To simulate an under-determined problem for the pendulum that would require control allocation, two cases are defined. In the nominal case, it is assumed that the pendulum is only using a torque on its hinge as the control action. In the second case, that will be called over-actuated case (to corresponds to the similarity of the case to the ICE aircraft) it is assumed that the pendulum is pulled by a magnet force to the right and therefore, two control actions of torque on the hinge and the force can be used for stabilizing the pendulum.

Next, the implementations of continuous/approximate dynamic programming methods of heuristic dynamic programming will be used on a nonlinear air-vehicle model to motivate the application of continuous methods to the ICE aircraft. Therefore, the ICE control problem will be compared to the nonlinear models of a rotatory pendulum and missile models in terms of nonlinearity and instability of the system and their respond to the RL controllers. The controllers have no knowledge about the optimal value function or policy and have to learn the optimal controller in batch and online modes.

The objective for the discrete controller is defined to move and keep the pendulum in an upwards position by implementing a torque at its hinge point or in the over-actuated case use the addition of the external force. For the air-vehicle model, an attitude tracking problem is defined.

5.2. Discrete Reinforcement Learning Implementation

The task of balancing the pendulum in the upright position can be seen as a regulation problem, which is close to the tracking problem in control applications. From another perspective, most of the tracking problems can be formulated as a regulation problem. Therefore, the swing-up task is chosen due to its low-dimensionality but at the same time, challenging and highly nonlinear control behavior. After the formulation of the problem in subsection 5.2.1, the system dynamics and equations of motion are presented in subsection 5.2.2. The simulation set up for experiments with the pendulum swing-up is described in subsection 5.2.4. Finally, the results of the discrete implementations are given by subsection 5.2.5 and subsection 5.2.6, respectively.

5.2.1. The pendulum swing-up and stabilization problem formulation

The swing-up pendulum problem is a classic control problem that also is fundamentally close to the control problem of rocket or missile propulsion (Kennedy and Conlon, 2011). Apart from that, other practical applications of this system have used in control literature. The system characterizes as an open-loop unstable and continuously falling over to reach equilibrium. Therefore, feedback control is required for balancing the pendulum in the upright position. Also, an external force or moment is needed to continuously interact with the state of the pendulum and correct its movements to avoid fall over. The PS problem in this study is separated into two categories: the determined pendulum problem, which will be referred to as a nominal problem, and the under-determined pendulum problem, which will be seen as an over-actuated problem. In the following, the dynamics for each one of these problems are individually discussed.

5.2.2. Nominal pendulum swing-up dynamics

As mentioned above, the first experiment includes the determined pendulum swing-up problem as its dynamics will be discussed here. The nominal case consists of two states of angle and angular rates, $s = \{\theta, \dot{\theta}\}$, and three possible actions (for discrete case) of clockwise torque, no torque, or a counterclockwise torque for the nominal problem. The maximum torque is defined in a way to avoid reaching the goal in one trial, and therefore, show the advantage of RL in a task not achievable by e.g., a proportional controller. The developed pendulum swing-up problem is demonstrated by Figure 5.1.

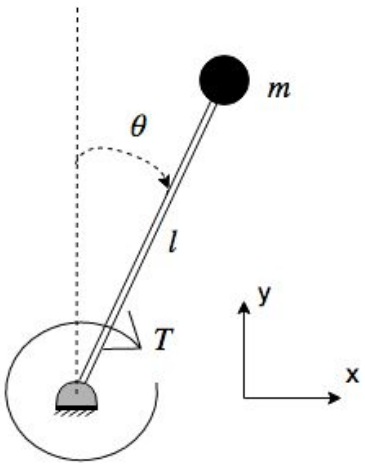


Figure (5.1) A schematic of the nominal pendulum swing-up problem, the arrows show the positive angular and torque directions.

The equation of the motion for this problem, derived from the Lagrangian of the system, is given as described by Equation 5.1.

$$ml^2\ddot{\theta} = -\mu\dot{\theta} + mglsin(\theta) + T \tag{5.1}$$

where the $\theta(t)$ is the angular position of the pendulum as presented by Figure 5.1, m is the mass of the pendulum in kg, l is the length of the pendulum in m, is the friction coefficient in kgm^2/s, g is the gravitational acceleration in m/s^2. The cable of the pendulum is assumed with no mass and inelastic. The state space is therefore defined in the following.

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}x_2\\ -\frac{\mu}{ml^2}\dot{\theta} + \frac{g}{l}sin\theta\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{ml^2}\end{bmatrix}T \tag{5.2}$$

This system will be used as the first experiment condition as described by subsection 5.2.4. Next, it is shown how this problem is modified to present an under-determined problem.

5.2.3. Over-actuated pendulum swing-up dynamics

As already discussed, this section includes the second experiment condition of the under-determined pendulum swing-up problem and its dynamics. The over-actuated pendulum swing-up problem includes again two states of angle and angular rates, $s = \{\theta, \dot{\theta}\}$. With the addition of the force F induced by the magnet, the number of possible actions for the discrete implementation increases to six and includes clockwise torque, no torque, or a counterclockwise torque, positive-x force, no force or negative-x force. The maximum torque and forces are defined in such a way to avoid reaching the goal in one trial to present the advantage of RL-based control. Figure 5.2 depicts a schematic of such a pendulum swing-up system.

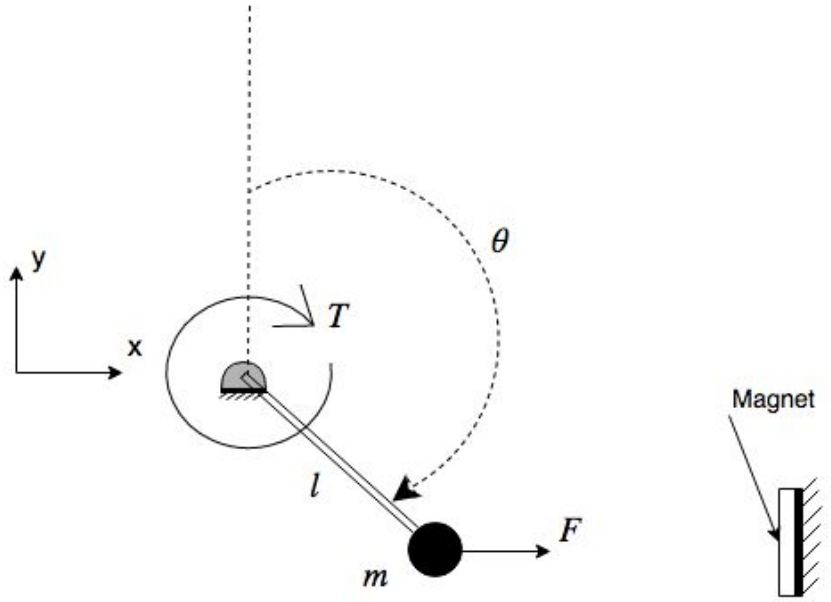


Figure (5.2) A schematic of the over-actuated pendulum swing-up problem, the arrows show the positive angular and torque directions.

With the equation of motions derived by Lagrangian of the system as:

$$ml^2\ddot{\theta} = -\mu\dot{\theta} + mglsin\theta + T - Flcos(\theta) \tag{5.3}$$

The state space of the over-actuated system is therefore defined as below:

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{\mu}{ml^2}\dot{\theta} + \frac{g}{l}sin\theta \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} T + \begin{bmatrix} 0 \\ \frac{-1}{ml}cos\theta \end{bmatrix} F \tag{5.4}$$

It can be seen that in such a system, the RL controller will have to perform a control allocation between the two control variables available, the torque and the force to the system, to position and balance the pendulum in its upright position.

5.2.4. simulation setup

The simulation experiments for the pendulum swing-up problem is performed in the MATLAB® version R2017b environment. Different RL controllers will be applied to the pendulum swing-up problem in a nominal and over-actuated modified case with the added force. The following properties are set for the pendulum problem in both cases of nominal and over-actuated: the mass of the pendulum m is $1.0kg$, the length of the pendulum l is $1m$, the friction coefficient is $0.01kgm^2/s$, and the gravitational acceleration g is $9.80m/s^2$ and as already mentioned the cable of the pendulum is assumed without mass. The transition from π to $-\pi\ rad$ will happen if the pendulum is positioned at the exact bottom. In the following by bottom, it is meant that the pendulum is at the angular position of $-pi$ or pi and top spot denoted to the angular position of zero. The control objective is defined as it is for a regulation problem: for the nominal problem, the torque input would try to make the pendulum reach the upright state and stabilize it there with the optimal policy, while in the over-actuated case both the force and the torque contribute to the regulation objective. This can be assumed as a classic control allocation problem, whereas the controller needs to distribute the use of available control powers since the number of effectors overpasses the number of control objectives. Although for an RL algorithm, over-actuation would not impose any fundamental difference in the approach, for a discrete implementation, it would cause the problem of high-dimensionality in storing the Q-function values, with increasing dimensionality by adding to the resolution of the states and actions. This motivates another benefit of employing function approximation methods.

For reinforcement learning algorithms with discrete state and action spaces, the state and action spaces need to be discretized. This means that a process is required for both states and actions before performing the algorithm and after each time step for ”rounding” the newly acquired states to the defined levels. Two state variables of θ and $\dot{\theta}$ are assumed to be symmetric and bounded with limits. The angle θ is discretized at thirty levels as $-\pi$, $-14\pi/15$, ..., $-\pi/15$, ..., $\pi/15$, ..., $14\pi/15$, and π. The angular velocity state $\dot{\theta}$ has also 30 levels of -10, -9.3103, ..., 9.3103 and $10rad/s$. The action space discretization was similar to the state space and will be separately discussed in subsection 5.2.5 and subsection 5.2.6.

The discrete Q-learning algorithm, which is a model-free approach, is chosen to be implemented for the pendulum swing-up problem. To accelerate learning and faster convergence, eligibility traces are used. The properties for Q-learning are chosen as $\alpha = 0.3$, $\gamma = 0.99$, and with $\epsilon = 0.1$ for greedy action selection and to assure exploration, particularly in the beginning of training. The ϵ value decreases for each episode by the amount of 1% (updated with a 0.99 factor) to improve exploitation and to assure the convergence of Q-learning to the optimal policy. In other words by having the $\epsilon : 0.1 \rightarrow 0.01$ the agent is more exploratory in the beginning and will become optimal in the later stages of the learning.

Each episode is started with a random angular position in the available state space of $[-\pi, \pi]rad$ and with a fixed zero angular velocity of $0rad/s$ as it is the condition when releasing the pendulum. A maximum angular rate of $10rad/s$ is defined for the pendulum, and the episode will be terminated if this maximum threshold is passed by the absolute value of the angular velocity. The simulation will run for 40,000 seconds, with each episode lasting for a maximum of 20 seconds. At the beginning of each episode, the states are reinitialized with a random rotational angle and zero rotational rate. The random selection will be made from the following subsets: $\dot{\theta} = [-10, +10]$ rad/s. The sampling

frequency is set to 100 Hz. This initialization is considered to characterize the condition where the pendulum is released from some point in the space with zero velocity, and the controller needs to find the optimal policy to bring the pendulum to the upright position which corresponds to zero angular position as depicted by Figure 5.1 and Figure 5.2. To account for randomness, the simulation can be repeated for each pair of random conditions. The low bounds of the torque are chosen so that the pendulum will need multiple swings to reach its goal. The state of the system is propagated for each time step using the simple Euler integration method as given by Equation 5.5.

$$x(t+1) = x(t) + \dot{x}.\Delta t \tag{5.5}$$

The continuous reward function is defined as an even function, as motivated by subsection 2.3.5 to be as described by Equation 5.6 for the discrete problem. The reward function gives the value of 0 if the pendulum is in the upward position, and for any other angular position, the reward value is defined negatively with a value of -1 if the pendulum falls over to the bottom point. In other rewords, the agent continuously receives punishment as it moves further from the upright position.

$$r(t+1) = cos(\theta) - 1 \tag{5.6}$$

The maximum amount of episodes is limited to 7000. This, however, does not mean that the agent might not have learned enough before this maximum number. To show if the agent has converged to a policy before the end of simulation episodes, the criteria for convergence is defined as Equation 5.8, which included the convergence of the Q-function and the policy. When the Q-function and policy convergence criteria are below a certain level ($c_1 and c_2$), the learning is terminated and considered complete, since the agent is not gaining any extra information and the results of the converge policy and Q-function will be shown. c_1 and c_2 might need to be defined separately.

$$Q - function\ convergence - criteria = \frac{\sigma|\Delta Q|}{\sigma Q} < c1 \tag{5.7}$$

$$policy\ convergence - criteria = \frac{\sigma|\Delta h|}{\sigma h} < c2 \tag{5.8}$$

Algorithm 1 shows the steps for the Q-learning implementation.

Algorithm 1 Q-learning control algorithm

Require: α, γ, ϵ, dt, episode number, maximum time step, state list, action list, c1, c2

Initialize $Q(s,a) \leftarrow 0$
Repeat for each episode:
  Initialize s
  Repeat for each step of episode:
    Updating a from s using the ϵ-greedy policy from Q
    Use action a to obtain next state s_p and the reward value r
    Update $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{\alpha'} {}_{Q(sp,ap)-Q(s,a)}]$
    $s \leftarrow sp$
    $a \leftarrow ap$
  $\epsilon \leftarrow \epsilon \times 0.99$;
  until angular velocity is within the of $[-10, 10]rad$
until $c1$ and $c2$ criteria is met

5.2.5. Q-learning for nominal pendulum swing-up problem

Since the focus of the thesis is on continuous RL application and discrete implementation will only be discussed here for comparison purposes, no benchmark on initial conditions or sensitivity analysis on the hyperparameters of the discrete controller is performed, unless it helps with understanding and motivating of the continuous RL approach. Instead, for approximate implementation for ICE, a comprehensive test benchmark will be developed. As already mentioned, the reward function provides the desirability of the states to the controller, and the researcher will design it. The reward function can include only the position of the system or also the actions if the control effort needs to be considered for

the system. The continuous reward function, as described by Equation 5.6, is used, which in addition to the observed state, is stored by the tabular Q-function values. The $\epsilon - greedy$ policy is used for selecting the next actions. The action space is discretized into three levels of torque values of $-5Nm$, $0Nm$, and $5Nm$. In the following the results for the application of the Q-learning on the SP with one available control input of torque is given.

Although the RL controller is able to find the optimal policy for a variety of situations, its full capabilities would not be demonstrated unless for a more complex task that needs to justify the instant reward to the long-term benefit. Therefore, first, a more complicated case is studied here, in which balancing the pendulum without gaining enough power by back and forth movements is not possible. To see the learning behavior of the controller, several variables are used, as will be discussed in the following. The median (ME) and moving average (MA) of the number of steps for which the controller was successful in balancing the pendulum when released from an arbitrary point in the space. Successful stabilization is defined when the angular position stays within ± 0.01 rad (± 0.5730 deg), and moving averages are obtained over each 100-time steps. Also, the cumulative reward collected per episode, the success rate, and regulation error (for performance indications) are used. An episode is assumed successful if the balancing task is achieved before 15 seconds (over 20 seconds of simulation time).

Figure 5.3 shows the result for learning and experimenting with batch data. It can be seen that from episode 200 onward the there is a large increase in the convergence steps while there is a slower increase in the number of steps that the agent is successful in stabilizing the pendulum afterward. After 1600 episodes, the changes become small. The same can be seen for the cumulative reward plot in the Figure 5.3(b). The reward also shows an overall increasing trend with some drops per some episodes while getting closer to zero. Since the initialization of the agent is random, it would take the agent some time to reach the balancing point, and since the rewards are count per time step, it is clear that the agent will never achieve exactly zero cumulative rewards, but it will get closer to it. The convergence criteria constants are set to $c_1 = 5^{-3}$ and $c_2 = 3.5^{-3}$ and the success rate achieved is 85% with tracking error of 0.005 rad.

The speed of the learning is much higher at the beginning of the training. It should be noted that the initial angular position is different at the beginning of each episode, and the agent is receiving value for the reward at each time step; therefore, it would be reasonable that the cumulative bonus would not converge to zero, but some negative value. Figure 5.3 presents the number of times that an element in the discretized state space is visited. In this figure, a color map is used with the mentioned color bar. As expected, the number of times the agent is close to desire stabilizing point or zero angular position and velocity is much higher, compared to the other states. It can also be seen that the agent met the bottom points quite a lot. This can be because the agent will try to make a push for backward and forward movements at the bottom to achieve enough power to reach the top. Therefore, it is expected that for the case where the agent is given a more straightforward goal, e.g., if the maximum torque is set higher, the agent would see the bottom line for a fewer number of times. The symmetry between the color map can be seen that shows that the trade-off for exploration is met to some reasonable level.

To see the best policy of the trained agent, next to the behavior of the controller for one trial, after the offline learning is completed, is shown by Figure 5.4. For the test trial, which shows the behavior of the trained agent in the control task, the pendulum will be started from the bottom position, and the agent will need to reach the upright position and stabilize there. It should be noted that the numbers of $\{-1, 0, 1\}$ in the y-axis of the graph corresponds to the torque values of $\{-5, 0, 5\}Nm$. For better presentation, the torque values are presented with a frequency of 0.1 seconds.

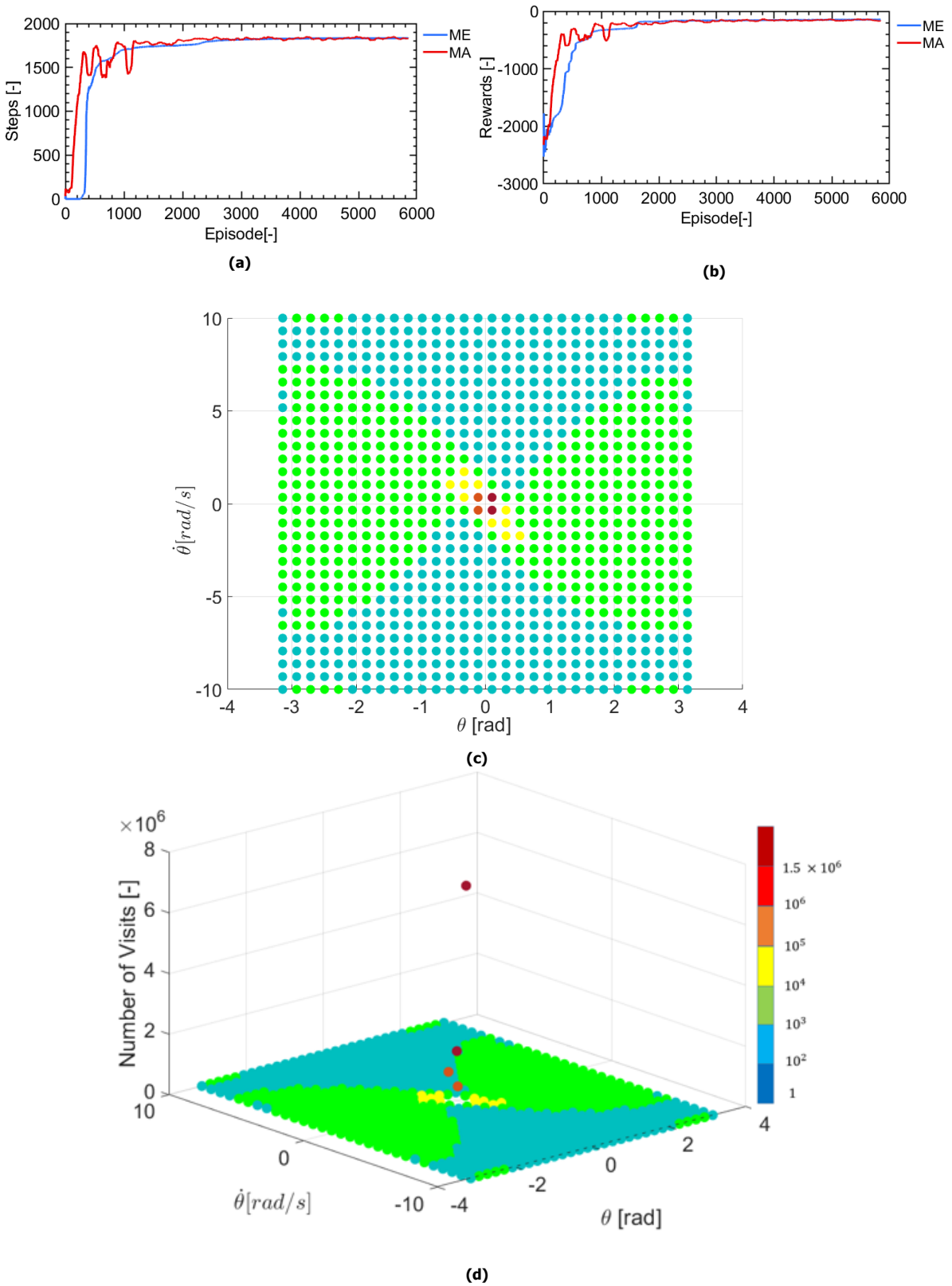


Figure (5.3) The results of Q-learning behavior for the *nominal* pendulum swing up problem for the (a) number of convergence steps, (b) cumulative rewards, (c) success rate and the over all learning number of steps visit from (d) perspective and (e) top view.

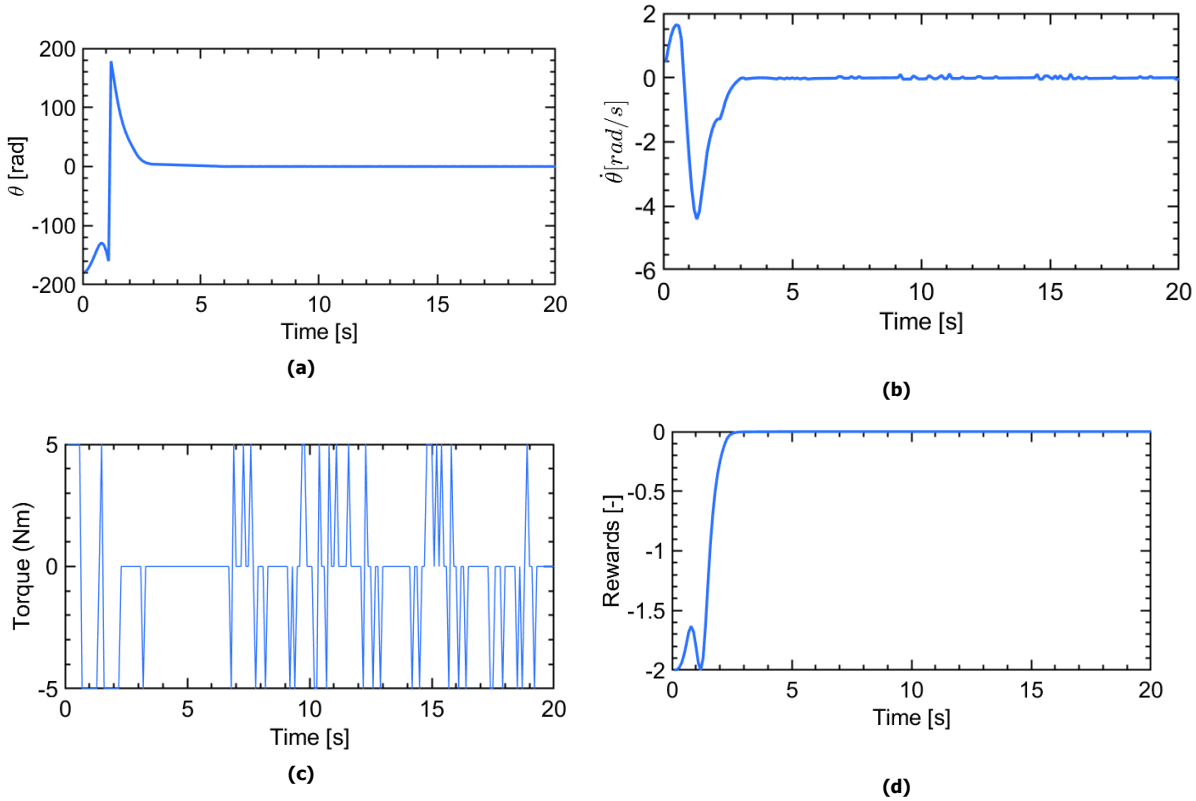


Figure (5.4) The results for the (a) angular position (b) angular rate (c) torque actions and (d) immediate rewards, obtained from the trained agent policy for the last representative trial for the *nominal* pendulum swing up problem.

Figure 5.4 shows that the agent can balance the pendulum after one back and forth movement at the bottom while both the angular position and rates converge to the zero values. The action input graph shows that there is a harmony between high and low ends of the available actions, while at some points, the agent chooses the zero input to keep the pendulum in the upright position. The plot showing the instant rewards also indicates that the agent can achieve the highest maximum zero reward after 4 seconds and, therefore, stabilizing the pendulum in its upright zero position. The optimal policy and the value-function values are sketched for the nominal problem in Figure 5.5.

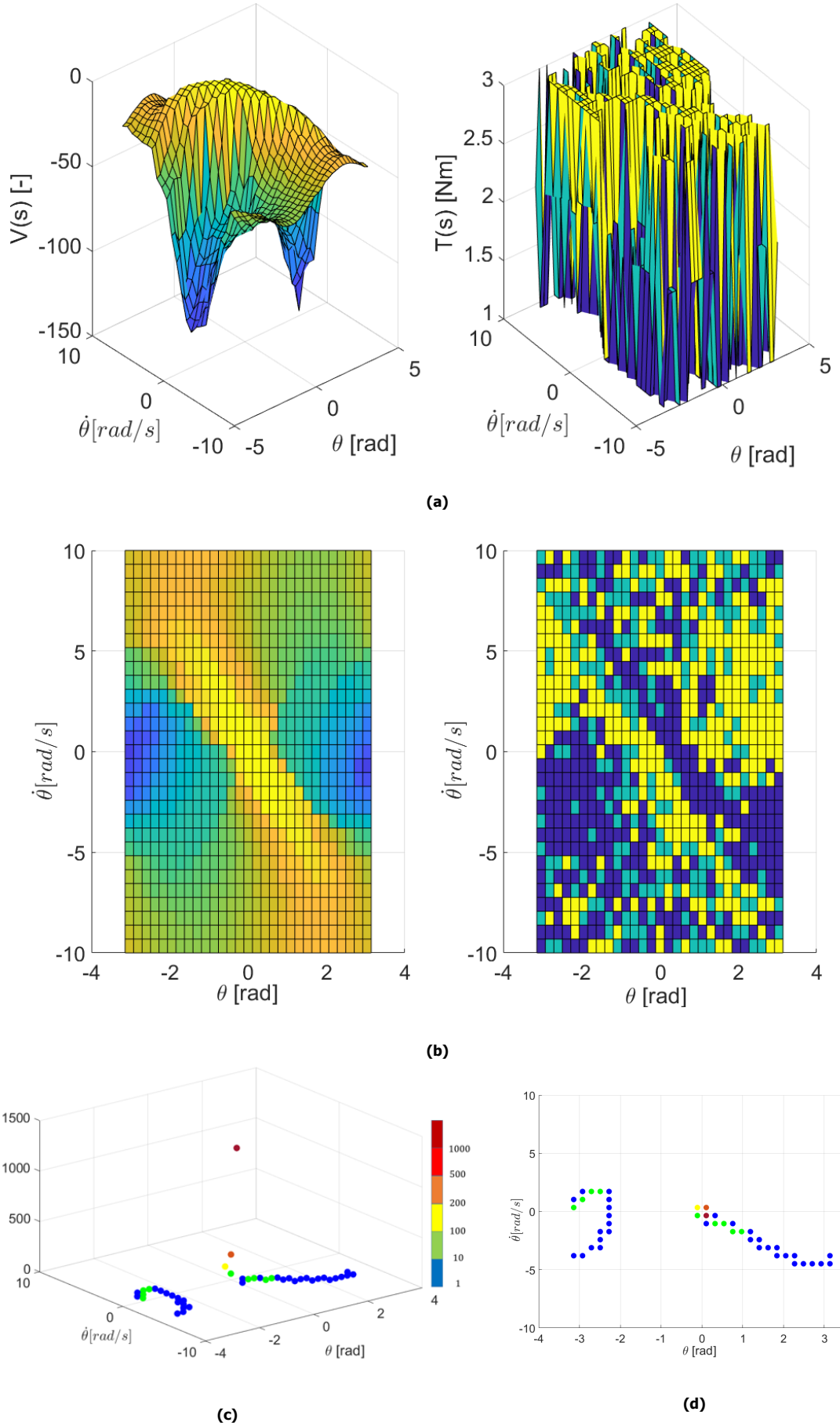


Figure (5.5) The optimal value-function (left graphs) and policy values (right graphs) in (a) perspective and (b) up views for the *nominal* pendulum swing-up problem in addition to the optimal trajectory for the (c) perspective and (d) top views.

In Figure 5.5, the $T(s)$ values of $\{1, 2, 3\}$ corresponds to the torque values of $\{-5, 0, 5\}Nm$. Therefore, the blue color in the right top and bottom graphs in the figure denotes zero torque. The yellow color denotes a maximum torque of $5Nm$, and the dark blue color denotes a minimum torque value of $-5Nm$. As can be seen from the graph, the inverted bell shape is obtained for the value function values, and so, the value function has higher values around the target point of the zero angular velocity. Some tendencies to the positive angular positions can be seen in the value function graph, that is due to the initialization of the training that caused this preference from the very beginning of the learning.

From Figure 5.5, it can also be seen that the agent has assigned high values for the bottom points but when the angular velocity is at its maximum in the opposite direction of the angular position. This can be because the agent is not capable of balancing the pendulum in one push. Therefore, it decides to locate the pendulum in states near the bottom where the angular velocity is in the opposite direction and its highest ends. Also, by looking at the training results for the policy, it can be seen that the agent is prone to use the torque in a positive direction when the agent angular position is negative, but the angular velocity is in a positive direction. Therefore, the agent helps the pendulum to reach the angular position. Also, for points were the pendulum is positioned close to the bottom and the angular velocity is close to zero or positive, the agent tries to add force in the positive direction while for some points with high angular rates it decide to apply no force or in the opposite direction to help not exiting the limit for the angular velocities.

On the other hand, by looking at the lower parts of the graph, one can see that for negative angular positions and negative angular rates, the agent is either applying no force or negative force in lower angular rates and positive force for higher negative angular rates. Therefore, either helping with back and forth movement or depriving the pendulum of exceeding the threshold for angular velocities. Also, when the agent is around the target balancing point, and the angular rate is highly negative the agent adds a torque in the same direction as velocity, but when passing to negative angular position, it will stop adding torque or adding positive torque to help the pendulum moving back to the upright position. The trajectory of the pendulum for the converged policy is shown by Figure 5.5 (c) and (d).

Another interesting case would be a situation in which the torque values were assigned in such a way that reaching the target point in one push is possible. Therefore, next, the results are shown for the case where the torque values can be chosen between constants of $\{-10, 0, 10\}NM$. The values for convergence criteria is set to $c_1 = 5 \times 10^{-3}$ and $c_2 = 2 \times 10^{-2}$. As Figure 5.6 shows, the controller for a simpler task can achieve higher successful steps compared to the more difficult task. This can be expected since the controller is now able to balance the pendulum in one go and do not need to come up with a complicated long-term strategy, and therefore, the jump in the number of convergence steps would happen with higher rates. In addition, the cumulative reward increases highly in the beginning and will continue increasing more slowly for the rest of the training episodes, this means that although the agent is exploring within the policies that result in achieving the goal with delays, but their cumulative reward is not necessarily lower than the policies with lower convergence rates.

By looking at the number of visits in state space for the training period, it can be seen that now, the controller will see the bottom end only for angular velocities close to zero. However, there is no more a harmonic increase in the number of state visits until the zero points is achieved. The agent, now visit states that have higher angular rates before reaching the zero angular position and close to zero velocity in the proximity of the balancing point. The success rate, in this case, is 91% with a tracking error of 0.0015 rad, which is less than the problem with a more difficult task. Figure 5.7, which depict the learning behavior, shows that in this more straightforward task of swing-up, a more harmonic and symmetric results are achieved for the value function, as the agent now do not necessarily need to perform any back and forth movement (i.e., proportional control is possible) to achieve the equilibrium at zero angular position. A smoother bell shape can be seen in this case, as depicted by Figure 5.7 (a). And the achieved policy shows some similarities with the complex problem while the zero torque seems to be used with less volume, and they have been replaced with maximum positive torque values. Therefore, the controller is more active in this case, and the control intensity is higher, as also depicted by Figure 5.8 (c). These constant changes in control actions between their limits can be dangerous for real-world problems. The converged trajectory can also be seen in this figure., which,

as can be expected, includes the push to the target balancing point and remaining there till the end of the episode period.

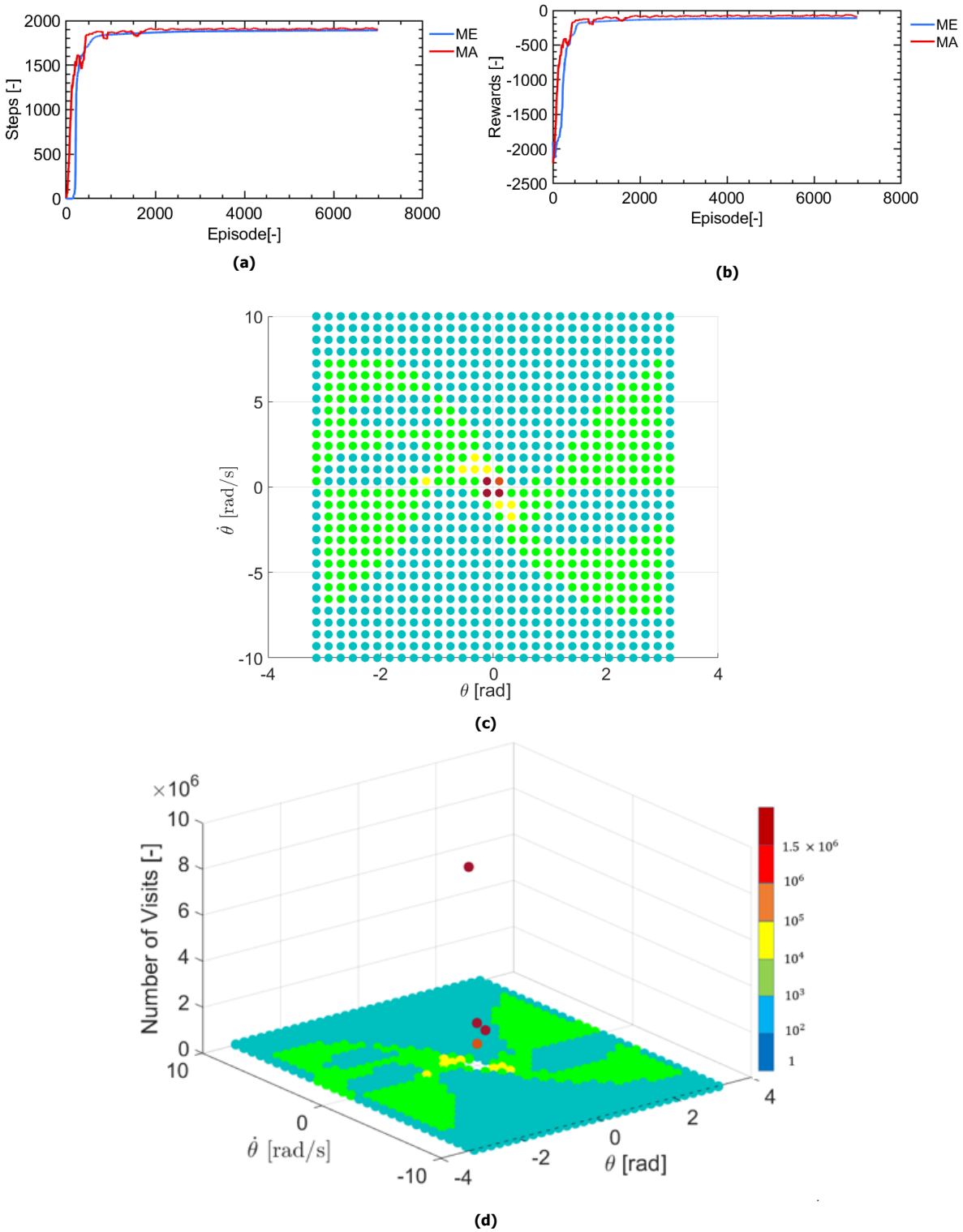


Figure (5.6) The results of Q-learning behavior for the *nominal* pendulum swing up problem with *simpler* task for the (a) number of convergence steps, (b) cumulative rewards, (c) success rates and number of state visits from (d) perspective and (e) top view..

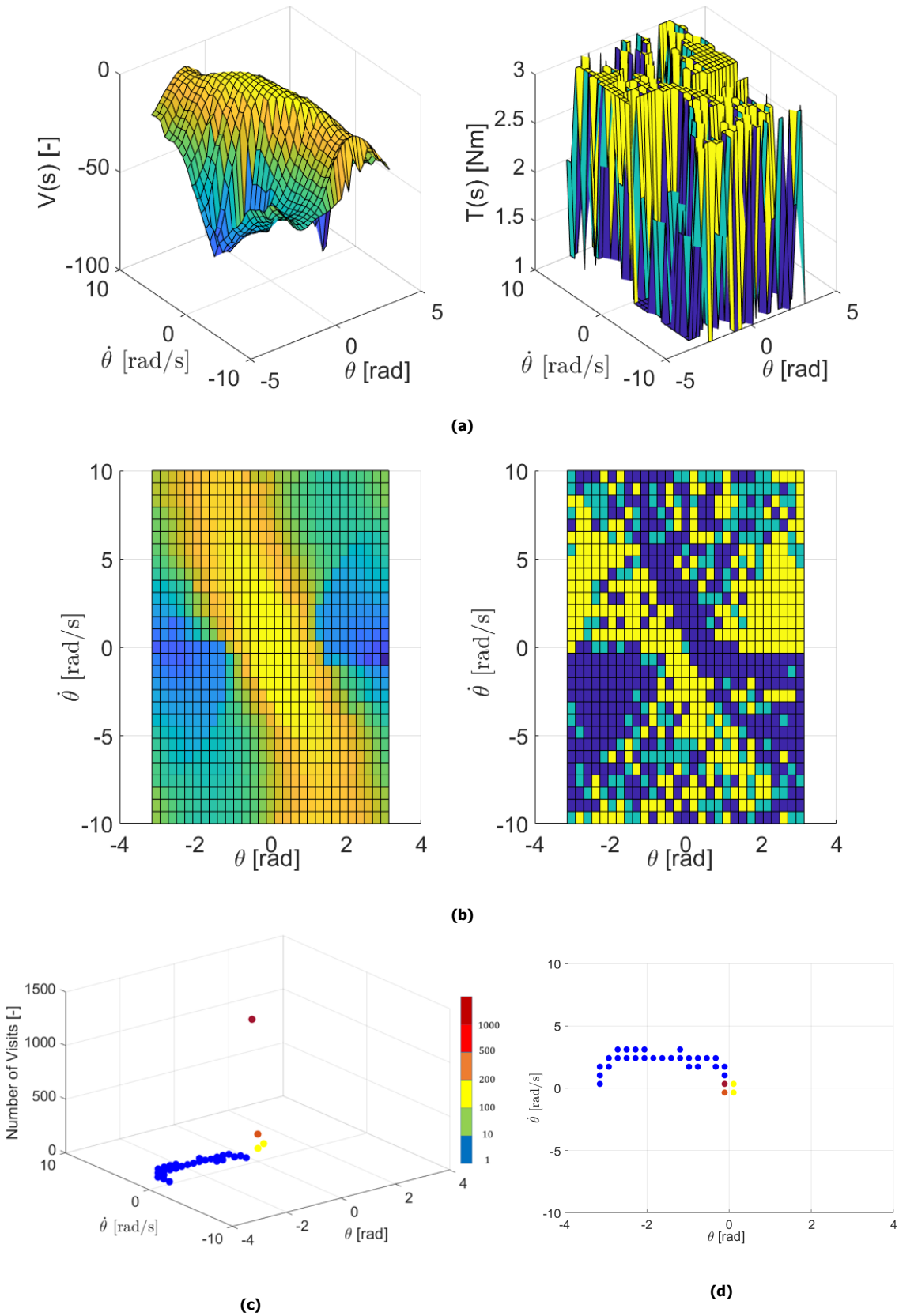


Figure (5.7) The optimal value-function (left graphs) and policy values (right graphs) in perspective and up views for the *nominal* pendulum swing-up problem with a *simpler* task in addition to the optimal trajectory in (a) perspective and (b) top view.

Looking at the trained agent behavior for the representative episode, the agent can reach the upright position in less than 2 seconds and will stay there for the remaining episode time. There are some variances and oscillations evident in the response of the angular velocity around the desired angular velocity of zero. It should be noted that agent in this experiment would not receive any punishment or reward for the angular rate values and the agent will try to stabilize the pendulum a zero angular rates based on the effect of the angular velocities in the pendulum angular position changes.

The significant variations in the torque values between actuators' limitations for this case can be seen by Figure 5.8 (c), as already mentioned above. This, for a real-life actuator, would not be suitable and not practical, considering the actuator delays. The reward values confirm the achievement of the desired angular position in the first 2 seconds, and it is remaining at the target value. The torque values are sampled with 0.1 seconds frequency for the converged episode.

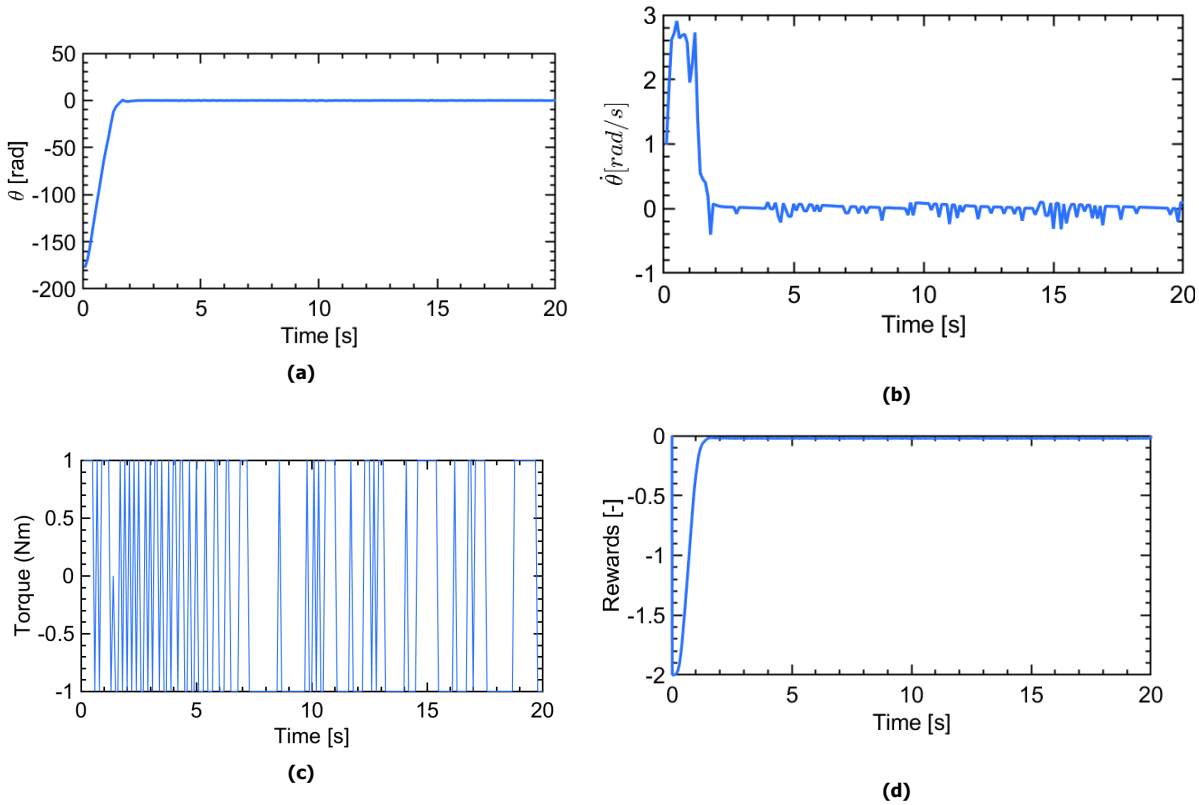


Figure (5.8) The results for the (a) angular position (b) angular rate (c) torque actions and (d) immediate rewards, obtained from the trained agent policy for the last representative trial for the *nominal* pendulum swing up problem with a *simpler* task.

5.2.6. Q-learning for over-actuated pendulum swing-up problem

As already mentioned for the over-actuated problem, the agent can learn the optimal policy in balancing the pendulum using two input actions of torque and an additional force. Therefore, the agent is solving an under-determined problem and have to come up with the optimal policy with redundancy in the control actions. This would result in a control allocation problem inherently obtained by the reinforcement learning solver. Q-learning with discrete state and action spaces is used for solving this problem. The $\epsilon-greedy$ policy is used as the mechanism for choosing the next actions. The maximum and minimum torque and force values are described in a way to mimic the maximum control power already used for the nominal case, to $[-2.5, 2.5]Nm$ and $[-2.5, 2.5]N$, respectively. While the state-space is discretized as mentioned by section 5.2, the action space for this problem is separated also into following levels $\{-1, -0, 1\}\times 5$ for both the torque and force input actions. Therefore, efforts have been made to use the same condition as control variables applied to both the nominal and over-actuated problem, although now more combinations are available.

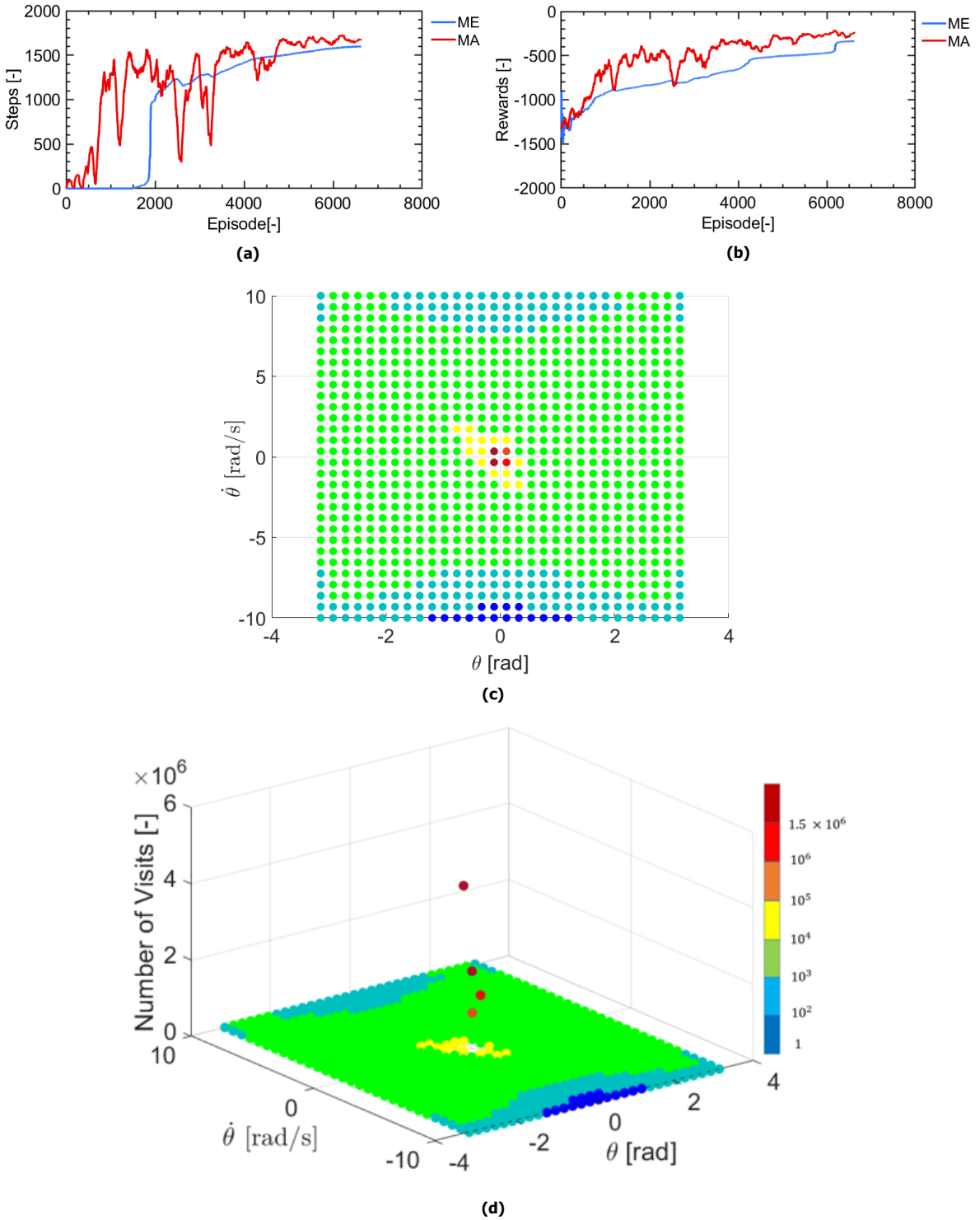


Figure (5.9) The results of Q-learning behavior for the *over-actuated* pendulum swing up problem.

Figure 5.9 shows the result for the application of Q-learning to the over-actuated swing-up pendulum problem. Looking at the moving average graph, it can be seen that although the increasing trend is evident in general for some convergence steps, success rate, and cumulative reward values, based on the median graphs, the speed of learning increases mainly only after 2000 episodes. In the case of the value of the cumulative rewards, the median of the sum of the rewards per episode increases monotonically. By looking at the graph for the number of state visits, it can be seen that most of the state space has been visited more than 1000 times, while the number of visits increases as the

system approach the desired angular position, and these visits become significantly high just around zero angular position and velocity. When compared to the nominal case, although the learning for the over-actuated example happens with more considerable variations, surprisingly, in 5000 episodes of training, the over-actuated agent can achieve a higher mean of success rates and also reaching the convergence of the state space faster. When looking at the state visits, it can be seen that in this case the states have been visited more frequently compared to the nominal case. Next, the converged value function and policy values are shown, in addition to the converged trajectory in Figure 5.10.

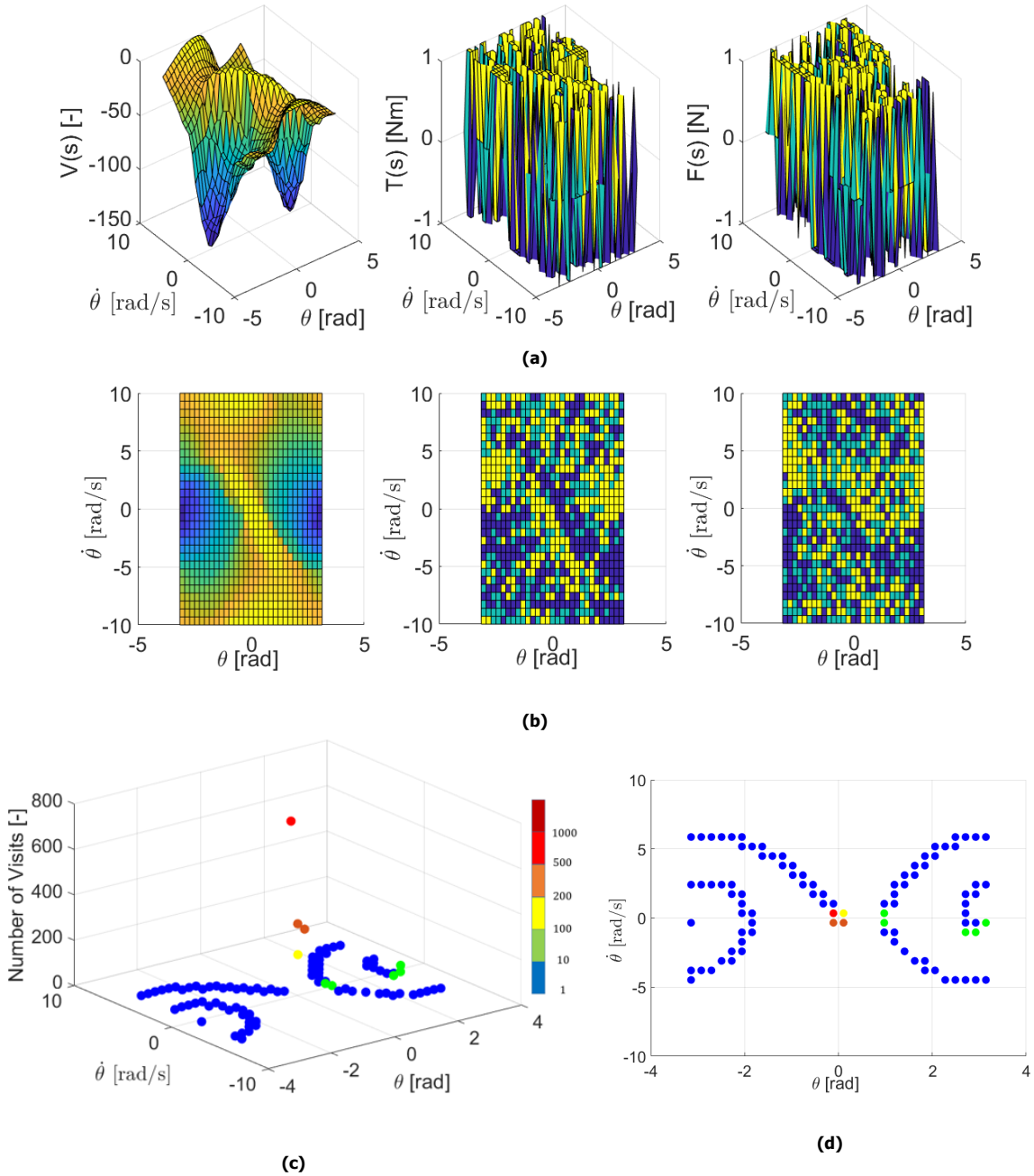


Figure (5.10) The optimal value-function (left graphs) and policy values (right graphs) in (a) perspective and (b) up views for the *over-actuated* pendulum swing-up problem in addition to the optimal trajectory for the (c) perspective and (d) top views.

By looking at the results for the policy and value function, the desired value-function shape is achieved. However, the agent, in this case, show also large values for more significant numbers of angular positions, including when the angular position exceeds 90 degrees. The values are high for these positions, only for cases that the angular rate is also high. The difference in the over-actuated case,

when compared to the nominal problem, is that in this case, the agent would give high angular rates in both directions for lower positions of the pendulum. For the nominal example, higher values were assigned to more significant angular velocities in the opposite direction of falling. However, in this case, the agent is taking the long-term rewards more into account by considering helping with falling to achieve more control power for directing upwards after passing the bottom line. Next, the learned policy is shown for the trained agent. The results are recorded with a frequency of 0.1seconds. The success rate for this case is 23% and tracking error 1.10 rad.

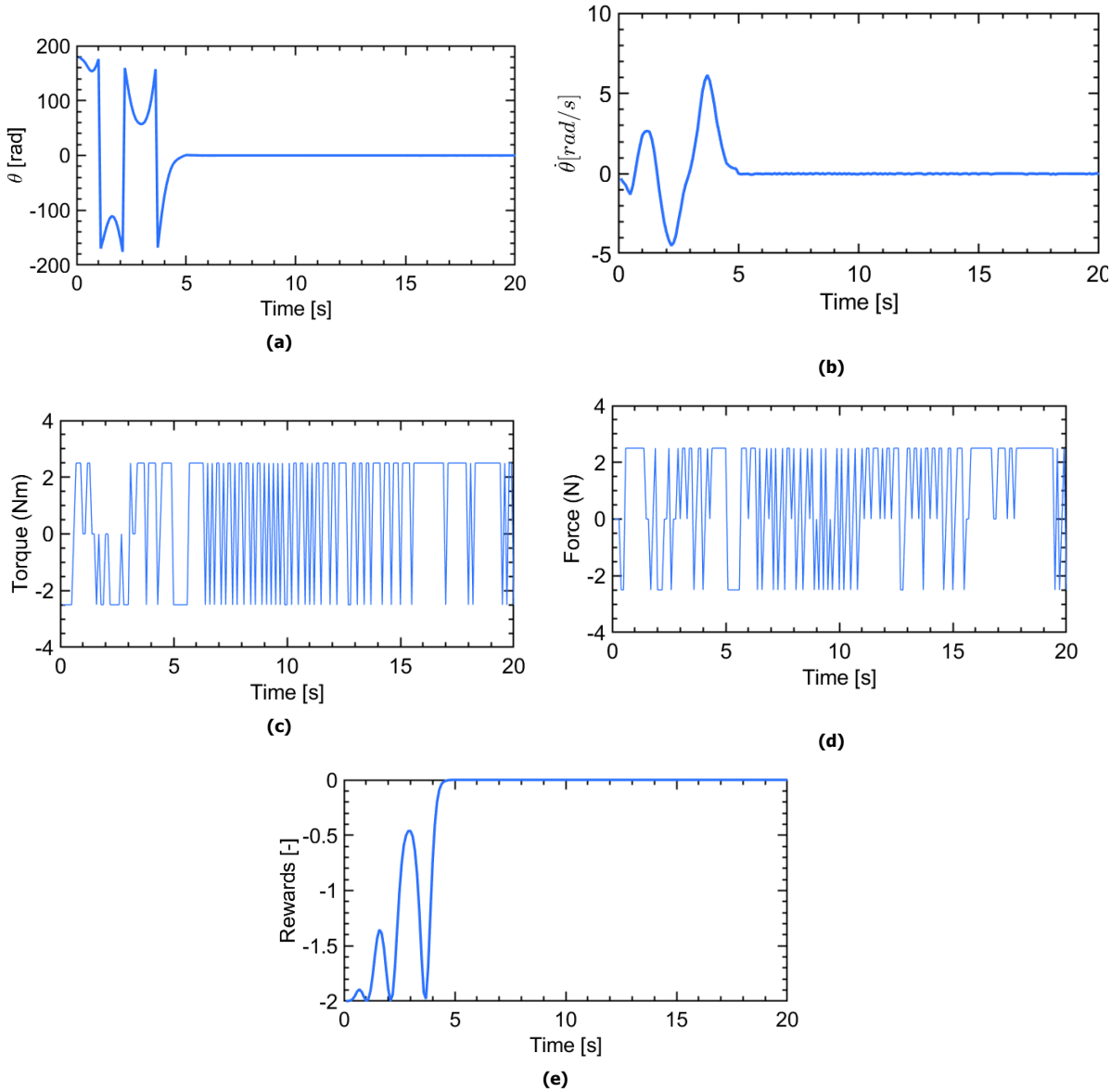


Figure (5.11) The results for the (a) angular position (b) angular rate (c) torque actions and (d) immediate rewards, obtained from the trained agent policy for the last representative trial for the *over-actuated* pendulum swing up problem for action space resolution of 1.

Figure 5.11 shows that the agent can balance the pendulum after 5 seconds, which shows a higher convergence time compared to the nominal problem. The agent, in this case, needs to solve a more complicated problem, and therefore, the delay can be expected. There can be seen some oscillations around the zero angular velocity. However, it should be noted that, as already mentioned, the angular velocity has not been included in the reward function for this case. Therefore, a benchmark can be done to study the effect of inclusion of the angular velocity in the reward function. In another attempt, the agent is given bigger system boundaries of $[-5\ 5]NM$ torque and $[-5\ 5]N$ force with the action scaling of $1Nm$ and $1N$ (mimicking the nominal case with $[-1010]NM$ available torque values). Now,

the agent is expected to balance the pendulum in one go. Figure 5.12 shows the results of the training.

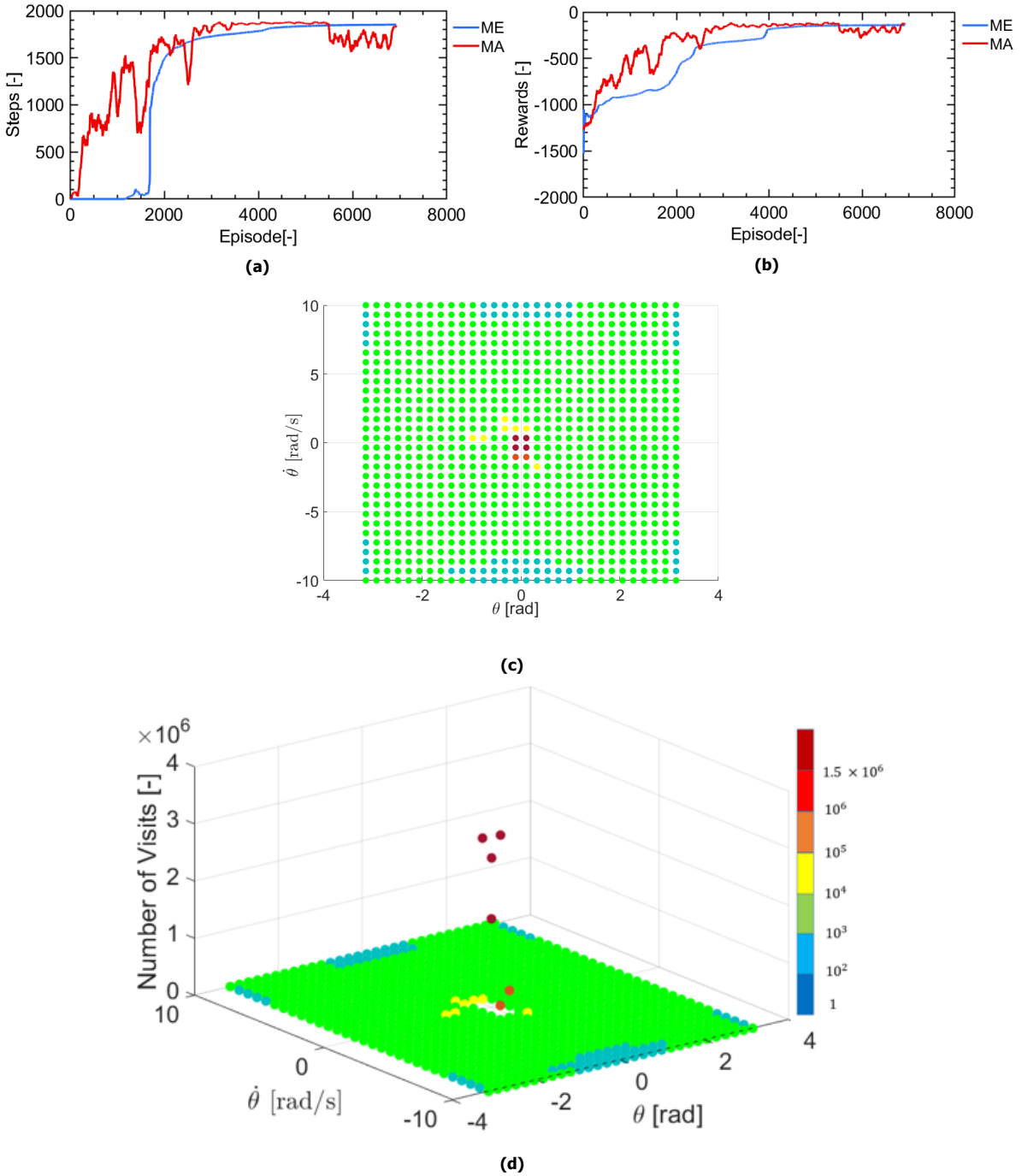


Figure (5.12) The results of Q-learning behavior for the *over-actuated* pendulum swing up problem with *simpler* task.

By looking at the learning graphs, one can see that the agent starts learning from the beginning of the training, which can be seen by the increasing trend of the convergence steps and rewards. However, looking at the medians, the agent increases to 2000 steps in a smaller number of episodes compared to a more difficult problem.

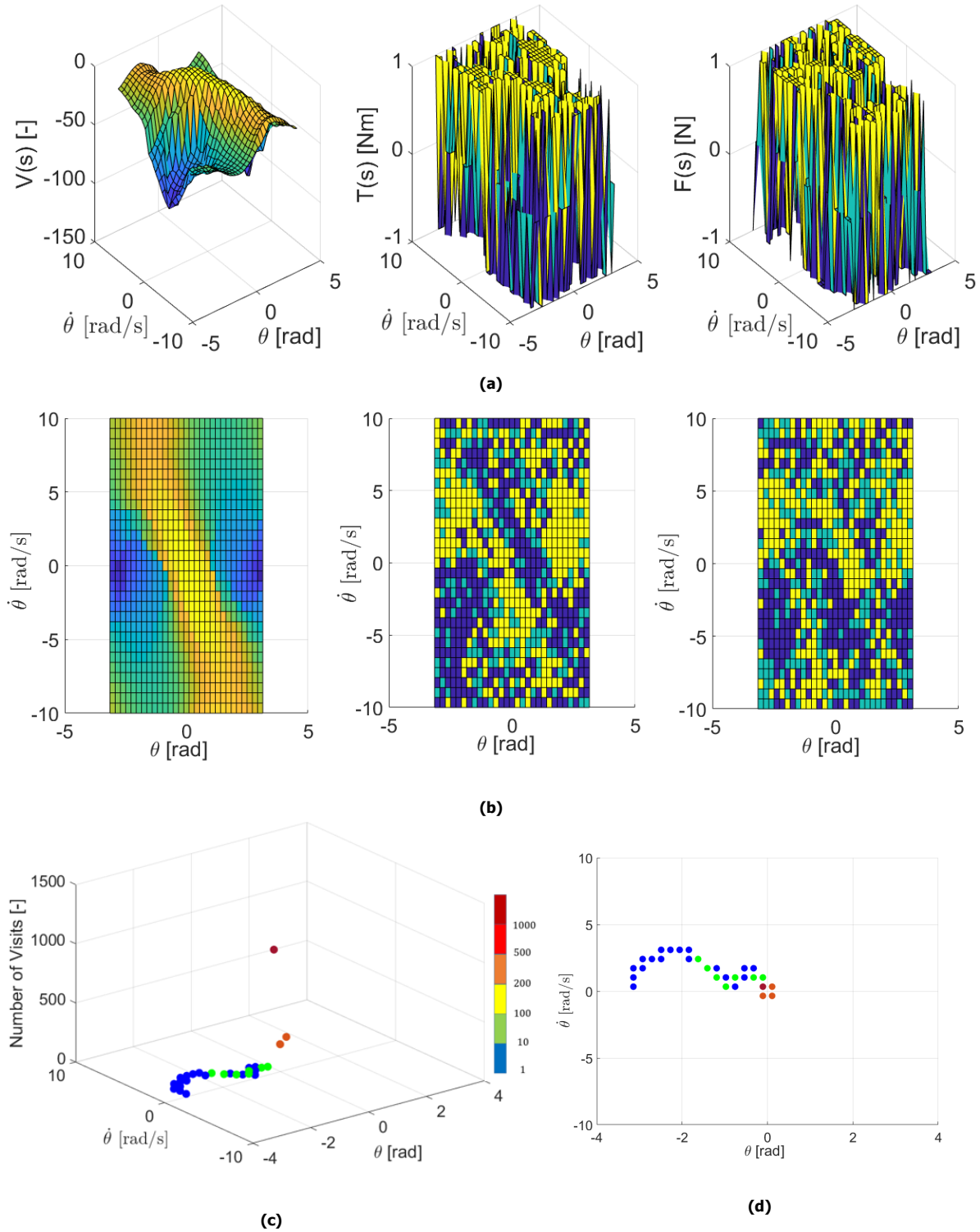


Figure (5.13) The optimal value-function (left graphs) and policy values (right graphs) in (a) perspective and (b) up views for the *over-actuated* pendulum swing-up problem with *simpler* in addition to the optimal trajectory for the (c) perspective and (d) top views.

Also, the agent shows higher visits for bottom angular positions with high angular rates, a behavior similar to the nominal case with a more difficult task. This can be interpreted as the effort of the agent to use one or both available action sources and it can be seen that it successfully use both force and torque values, as it would result in stabilization of the pendulum in one swing and that is why fewer values are given to these points as shown by Figure 5.13. Therefore, the agent can achieve the

balancing point without moving back and forth as can be seen by the trajectory points.

The trained agent behavior is shown by Figure 5.14. Similar to the nominal problem, the pendulum will start from the bottom for the representative trial, and the controller will need to balance the pendulum with the best combination of the torque and forces. The results are shown with a resolution of 0.1 seconds.

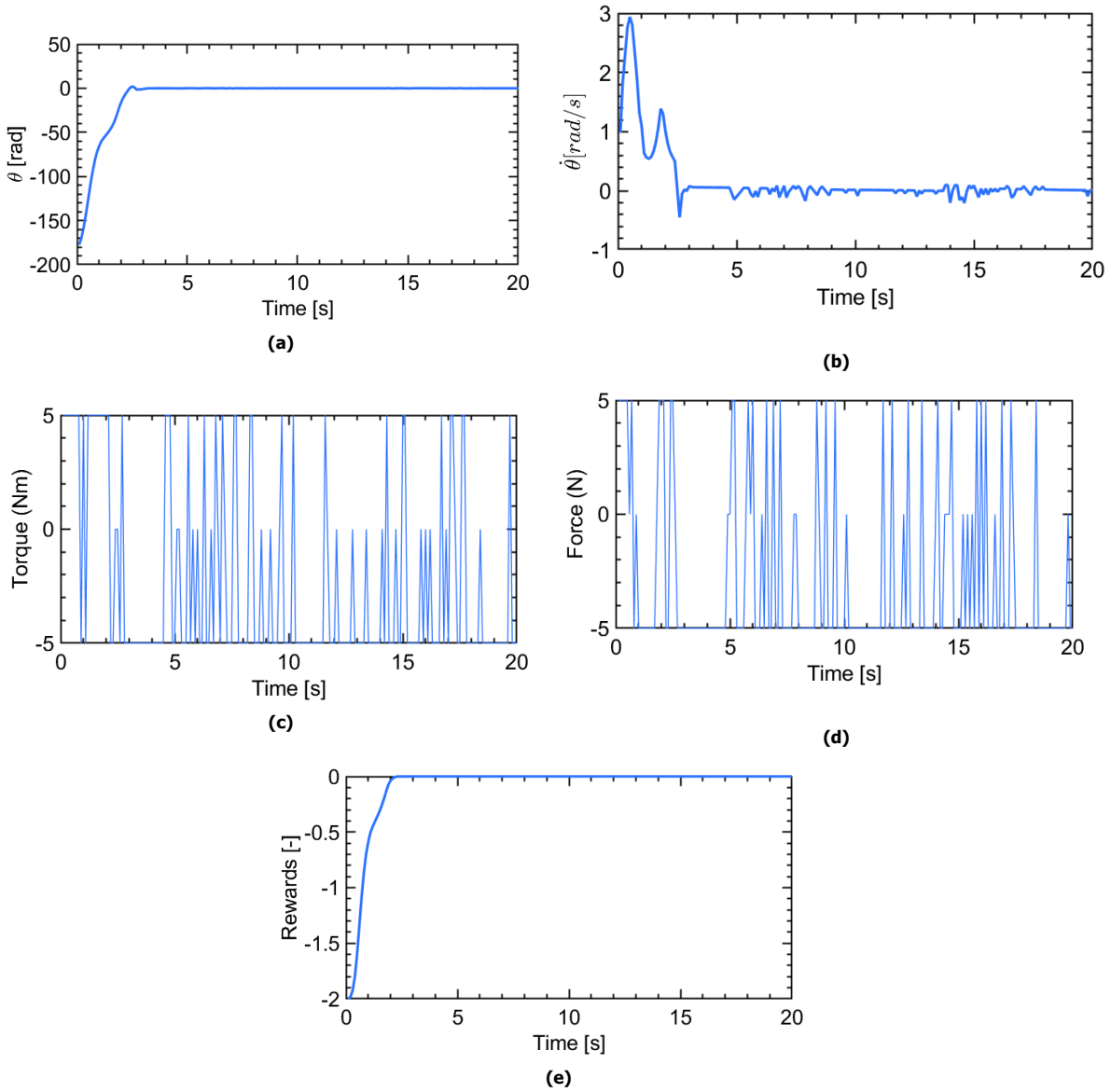


Figure (5.14) The results for the (a) angular position (b) angular rate (c) torque actions and (d) immediate rewards, obtained from the trained agent policy for the last representative trial for the *over-actuated* pendulum swing up problem with *simpler* task for action space resolution of 1.

The graph shows that the controller can balance the pendulum in about 2 seconds. For the nominal problem with the simpler task, the agent was able to adjust the pendulum in about 1 second. However, by looking at the history of the actions taken by the agent for the swing-up, it can be seen that the torque actuator is continuously active and moves between the limits. This is not the case for the over-actuated problem, where the control powers are divided between the two available actions. Therefore, less control intensity can be seen by just looking at the graphs. The success rate for this case is 26% and tracking error 0.294 rad.

It should be noted that there is no penalty for the use of different action value levels in this prob-

lem. Therefore, the agent would not seek less control intensity. The reason is that the study followed the suggestions made by literature for the definition of the action space. To see whether the agent benefits from receiving penalties for more extensive control power usage, a benchmark can be studied for this case (not studied in this report).

5.3. Continuous Reinforcement Learning Implementation

One deficiency with discrete RL algorithms is that there is a little transfer of learning experiences between different situations. To expand the RL problem to the real world large problems, the continuity of the state and action space should be handled by the RL controller. From chapter 2 and chapter 3, it was found that approximate dynamic programming methods that can deal with nonlinear systems are promising for this thesis project. One of the popular structures of approximate dynamic programming is the heuristic dynamic programming, which can generate an adaptive near-optimal controller (online) with no prior knowledge on the system dynamics. Therefore, in this section, the conventional HDP, which uses a nonlinear global model of the system as described by section 2.5, is used to show the validity of continuous approaches in a nonlinear control problem of missile flight control. Neural networks are used as the function approximation method in the learning phase as they produce approximate functions that are smooth, differentiable, and locally sensitive.

The goal is to gain experience with the implementation of ADP methods in general and also show their pros and cons, e.g., compared to the classic discrete RL solutions such as Q-learning, which was discussed in the previous sections. Therefore, this section first describes the dynamics and tracking control problem of the missile system. Next, the HDP method application on the tracking problem is given in an online framework without persistent excitation and with the added PE to the learning to satisfy the exploration requirement for the RL controller at the beginning of the learning. The results of the experiment are given for each condition separately.

5.3.1. Missile flight control dynamics

In this section, a simplified nonlinear missile model for tracking a reference signal is introduced, based on the Kim *et al.* (2004) and Zhou (2018). The missile model is a second-order continuous, highly nonlinear model while being simple to eliminate the need for offline learning. The short period model consists of two states, the angle of attack α and pitch rate q. The tracking task only includes the pitch rate tracking with the elevator deflection δ_e as the control input. The nonlinear model is shown by **??** in the pitch axis.

$$\dot{\alpha} = q + \frac{\bar{q}S}{m_\alpha V_T} C_z(\alpha, q, M_\alpha, \delta_e), \tag{5.9}$$

$$\dot{q} = \frac{\bar{q}Sd_l}{I_{yy}} C_m(\alpha, q, M_\alpha, \delta_e) \tag{5.10}$$

where α is the angle of attack, q is the pitch rate, $\bar{q}$ is the dynamic pressure, S is the reference area, m_α is the missile mass, V_T is the speed, C_z is the force coefficient in the body Z-direction, M_α is the Mach number, δ_e is the elevator deflection, d_l is the reference length, $I_y y$ is the pitching moment of inertia and C_m is the pitch moment coefficient. In the above equations, C_z and C_m are nonlinear functions given by:

$$C_z(\alpha, q, M_\alpha, \delta_e) = C_{z1}(\alpha, M_\alpha) + B_z \delta_e, \tag{5.11}$$

$$C_m(\alpha, q, M_\alpha, \delta_e) = C_{m1}(\alpha, M_\alpha) + B) m \delta_e, \tag{5.12}$$

$$\tag{5.13}$$

where

$$\begin{aligned} B_z &= b_1 M_\alpha + b2, \\ B_m &= b_3 M_\alpha + b_4, \\ C_{z1}(\alpha, M_\alpha) &= \phi_{z1}(\alpha) + \phi_{z2} M_\alpha, \\ C_{z2}(\alpha, M_\alpha) &= \phi_{m1}(\alpha) + \phi_{m2} M_a lpha, \\ \phi_{z1}(\alpha) &= h_1 \alpha^3 + h_2 \alpha|\alpha| + h_3 \alpha, \\ \phi_{m1}(\alpha) &= h_4 \alpha^3 + h_5 \alpha|\alpha| + h_6 \alpha, \\ \phi_{z2} &= h_7 \alpha|\alpha| + h_8 \alpha, \\ \phi_{m2} &= h_9 \alpha|\alpha| + h_{10} \alpha, \end{aligned} \tag{5.14}$$

where these aerodynamic parameters only valid for angle of attack range of $\alpha \in [-10, 10] deg$ and M_α is constant and equal to 2.2 and $b_1, ..., b_4, h_1, ..., h_{10}$ are the constant coefficients identified in the flight envelop.

5.3.2. Simulation setup

Same as the discrete approach, the framework is programmed in MATLAB® R2017b. The control objective is set for the controller to follow an arbitrary reference signal for the angle of attack over an episode, which lasts for 40 seconds. Therefore, learning happens online. The reference signal is defined as a sinusoid with a constant amplitude and frequency as described by:

$$\alpha_{ref} = 5sin(t) \tag{5.15}$$

The explicit quadratic cost function for the problem is defined with an arbitrary weighted matrix multiplied by the squared of the tracking error for the pitch rate as given by:

$$c(t) = Q_c(\alpha - \alpha_{ref})^2 \tag{5.16}$$

where Qc is given as:

$$Q_c = \frac{C}{\alpha_{max}^2} \tag{5.17}$$

where α_{max} equals $10 deg$ and C is a constant. As illustrated by section 2.5, the HDP algorithms use two steps of policy evaluation, carry out by the critic and policy improvement, done by the actor to get close to the optimal policy while the model of the system will be identified, in this case online. For this experiment, the critic, actor, and model are approximated with Artificial Neural Networks (ANN) as described by section 2.4. The output layers are assumed linear, and the hidden layers activation functions are given by continuous and nonlinear hyperbolic tangent function as:

$$\phi(h) = tanh(0.5h) \tag{5.18}$$

This same setting is applied for the actor, critic, and model networks. The number of hidden layer neurons for the actor, critic, and model is set to 12. The weights of the NN are updated iteratively with gradient descent method using Least Mean Square (LMS) described by section 2.4. To avoid the ”explosion” of the neural network weights, the weights will be limited in the range of [-30, 30], the approach also followed by Zhou (2018). The critic and actor both receive the augmented state, which includes the angle of attack tracking error. Therefore, the networks can detect the correlation with the cost function easier as shown by literature in case of a linear flight control problem Kroezen (2019) and the nonlinear missile tracking Zhou (2018). The resulting state vector is $x = [\Delta\alpha]$ and the control input vector of the system is $\mathbf{u} = \delta_e$. The model parameters are shown in Table 5.1.

Table (5.1) Nonlinear missile model parameters

Model parameter	unit	Value	Model parameter	unit	Value
M_a	-	2.0	b_4	-	-48.2246
g	ft/sec^2	32.2	h_1	-	-288.7
W	lbs	450	h_2	-	50.32
V	ft/sec	3109.3	h_3	-	-23.89
I_{yy}	slug ft^2	182.5	h_4	-	303.1
q	lb/ft^2	6132.8	h_5	-	-246.3
S	ft^2	0.44	h_6	-	-37.56
dl	ft	0.75	h_7	-	-13.53
b_1	-	1.6238	h_8	-	4.185
b_2	-	-6.7240	h_9	-	71.515
b_3	-	12.0393	h_{10}	-	10.01

The convergence of the critic, actor, and the model networks are, to a high degree, dependent on the learning rate of the weight updates. The improper choice for the learning rate α can cause the controller to be trapped in a local optimum or oscillate around the optimal weights. To decrease the sensitivity of the HDP networks to the initial learning rates, the adaptive learning rates can be used. The approach includes self-tuning of the learning rates for each time step by multiplying the current learning rates by 0.9, 0.8, and 0.99 for the critic, actor, and model learning rates, respectively. This update holds until the learning rates are bigger than 0.002 for the critic and the actor and bigger than 0.001 for the model learning rates. The updated learning rate will be used for the next time step. The adaptive learning rates apart from their effect on training acceleration can eliminate the requirement of learning rate schedule. The HDP learning parameters are given by Table 5.2 with the learning rates assigned at the beginning of the learning. These values are obtained empirically and can be changed to achieve better performance by doing sensitivity analysis.

Table (5.2) HDP framework learning parameters for the case with no PE and with PE

Learning hyper-parameter	symbol	Value (no PE)	Value (PE)
time step	Δt	0.001	0.0005
initial critic learning rate	α_c	1	1.5
critic discount factor	γ_c	0.99	0.99
initial actor learning rate	α_a	500	0.01
initial model learning rate	α_m	15	1.5
cost function constant	C	200	100

5.3.3. Persistent excitation

For any RL method, the same for ADP methods, evaluation of the policy by the actor depends on the exploration of the state-space, known as Persistent Excitation (PE). For aircraft models, there are techniques to excite the aircraft modes, such as sine waves, doublets, 3211 doublets, and pseudo-random noise. The general dynamic of the system with PE in terms of state equations can be given as:

$$\dot{x}(t) = f[x(t), \mathbf{u}(t) + \mathbf{d}(t)] \tag{5.19}$$

, where x is the state vector, $\mathbf{u}$, is the control input vector, and $\mathbf{d}$ is the input disturbance that for this project, it is set to be caused by the input noise. This disturbance is defined as a filtered Gaussian noise and is only implemented for the first 10 seconds of online learning. The purpose is to add exploratory actions to the actor network values. The exploration is required for the identification of the optimal value function. The PE will persistently excite the system for online identification of the model and exploration of the state-space. As disturbance is not desirable in the real-world applications, the RL controller needs to do the task of tracking and stabilization of the system in case of disturbance. The HDP methods are shown to be capable of compensating for the disturbances. The action disturbance uses Equation 5.3.3 to generate exploratory actions added to the actor network output.

$$d(t) = \sigma N(t) \tag{5.20}$$

, where $N(t)$ is the Gaussian noise and σ, is the noise amplitude. For this experiment, σ is set to 5.

5.3.4. Implementation

The HDP implementation is shown by a pseudo-code by Algorithm 2. The actor, critic, and the model networks are initialized with random weights between $[-1, 1]$ and will be updated for each time step recursively. The critic outputs for each time step will be performed for the previous states, current states, and the next states obtained from the model with the current weights of the critic. The critic then is updated based on the current states while the actor is updated with the next cost-function estimated by the critic.

Algorithm 2 HDP control algorithm

Require: algorithm parameters: αs, γ_c, dt, state-space boundaries: s_{max} and inputs: $\hat{J}(s, \mathbf{W_c})$, $\pi(a|s, \mathbf{w_a})$, $M(s, a, \mathbf{w_a}), Q_c$, Q_m and J^*

Initialize the actor, critic and the model network weights randomly $\mathbf{w_{c_0}}, \mathbf{w_{a_0}}, \mathbf{w_{m_0}} \in [-1, 1]$
Initialize $s \leftarrow 0$
if persistent excitation is used **then**
    Initialize noise parameters Pe_0, n_c, τ, J_{min} and J_0
repeat for each step of episode (last for 20 seconds simulation time):
    Update s_{ref} for $t-1$, t and $t+1$ to obtain augmented state s^a for $t-1$, t and $t+1$
    $\hat{J}(t) \sim \hat{v}(s^a(t), w_c)$
    $a \sim \pi(.|s^a(t), \mathbf{w_a})$
    if persistent excitation is used **then**
        $a \leftarrow a + n(t)$
    $s_{next} \sim M(s^a, a, \mathbf{w_m})$
    $\hat{J}(t-1) \sim \hat{v}(s^a(t-1), \mathbf{w_c})$
    $\hat{J}(t+1) \sim \hat{v}(s^a(t+1), \mathbf{w_c})$
    $r \leftarrow Q_c[s^a(t-1)]^2$
    $e_a \leftarrow J^* - \hat{J}(t)$
    $\mathbf{w_a} \leftarrow \mathbf{w_a} - \alpha_a e_a \nabla_{\mathbf{w_a}} \pi(s^a, \mathbf{w_a})$
    $e_c \leftarrow r + \gamma_c \hat{J}(t) - \hat{J}(t-1)$
    $\mathbf{w_c} \leftarrow \mathbf{w_c} - \alpha_c e_c \nabla_{\mathbf{w_c}} \hat{J}(s^a, \mathbf{w_c})$
    $s(t+1) \leftarrow plant\_dynamics(s, a)$
    $e_m \leftarrow s(t+1) - s_{ref}(t)$
    $\mathbf{w_m} \leftarrow \mathbf{w_m} - \alpha_m (Q_m + Q_m^{'}) e_m \nabla_{\mathbf{w_m}} \hat{M}(s^a, a, \mathbf{w_m})$
    if $\alpha_c \geq 0.002$ **then**
        $\alpha_c \leftarrow \alpha_c \times 0.9$
    if $\alpha_a \geq 0.002$ **then**
        $\alpha_c \leftarrow \alpha_c \times 0.8$
    if $\alpha_c \geq 0.001$ **then**
        $\alpha_c \leftarrow \alpha_c \times 0.99$

In the following, the results are shown for the HDP implementation on the missile angle of attack tracking problem.

5.3.5. Results for heuristic dynamic programming for nonlinear missile model

The Heuristic Dynamic Programming controller is used to control the nonlinear missile model with online model identification. The identified model is used for predict the next states $\hat{x}_{t+1}$ that later will be used for cost function estimation for the next time step $\hat{J}(\hat{x}_{t+1})$. The difference between this value from J^* is used for actor training. A total of 40000 time steps with a sampling time of 0.0001 seconds, so 40 seconds of simulation time (about seven periods of the reference signal), is chosen for online learning. In the following, the performance of the controller is discussed along with its stability.

For this experiment, the performance of the HDP controller is measured based on the tracking performance and the cost function values per time step in online learning. Also, the critic, actor, and model learning errors are presented. Using tracking performance, the controller behavior can be observed, and the cost function values over time show if the controller is moving in the direction of minimizing the cost. Initially, the results showed a lot of instability since the controller is not satisfying the PE condition. Therefore, to excite it, larger initial random weights are chosen, so between [-1, 1], in addition to high initial learning rates, as given by Table 5.2. The results are shown for the successful trial.

Figure 5.15 shows the results for the HDP controller with no PE. As one can see from the figure, the HDP controller can follow the given reference trajectory signal over time in online learning. At the

beginning of the learning, the response, although smooth, shows less good tracking. As the learning progresses, the response becomes closer to the reference trajectory.

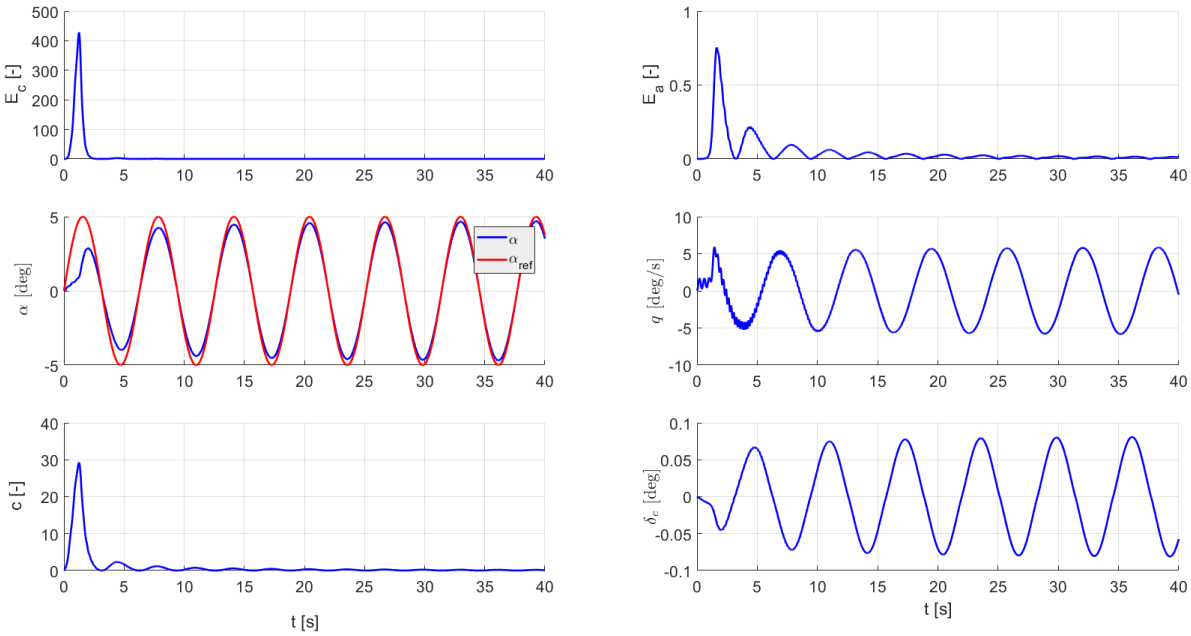


Figure (5.15) Results of nonlinear missile model with HDP controller for critic error (top left), actor error (top right), angle of attack tracking (middle left), pitch rate response (middle right), cost values (bottom left) and elevator deflection (bottom right)

The response is smooth, and no sudden deflections can be seen after the maxima and minima of the reference signal, while the reaction of the angle of attack becomes closer to the reference signal and the pitch rate response become smoother. The commanded deflection is also smooth over the learning, even at the beginning of the learning, where the pitch rate response is not smooth, and there is some inconsistency before the first 3 seconds. This shows the benefit of continuous methods compared to discrete methods showed previously in this chapter, where the control commands were bang-bang and so the actuators would push to their limits. By looking at the actor and critic errors in the same graph, one can see that over the episode, first, there is a significant error in both cases. This can also be seen by the large cost value before 3 seconds of learning. Cost per time step is a useful metric for evaluating the performance of the controller, as the HDP approach is expected to minimize this value during online learning. It can be seen that, although at the start of the episode, there are some spikes as the agent learns the control policy, later in the learning, the error values and cost get close to the zero. There are still some small oscillations that are due to the tracking error in the maxima and minima of the reference signal.

The online model learning performance is shown by Figure 5.16. The angle of attack is identified immediately by the model network, while for the pitch rate, the model shows some inconsistency before the first 10 seconds of learning. However, as the episode continues, the model identifies the pitch rate response very closely. The model error is also shown in the same figure. The error here is defined by:

$$E_m = (Q_m + Q'_m)^2 e_m \tag{5.21}$$

where

$$Q_m = 2\begin{bmatrix} \frac{1}{\alpha_{max}^2} & 0 \\ 0 & \frac{1}{q_{max}^2} \end{bmatrix} \tag{5.22}$$

$$e_m = (x_{plant} - x_{model})$$

As one can see, the model error before 10 seconds is higher (although quite small), which can be explained by the pitch rate model errors that are presented in the overall model error. After 10 seconds, the model error becomes small and converges to zero.

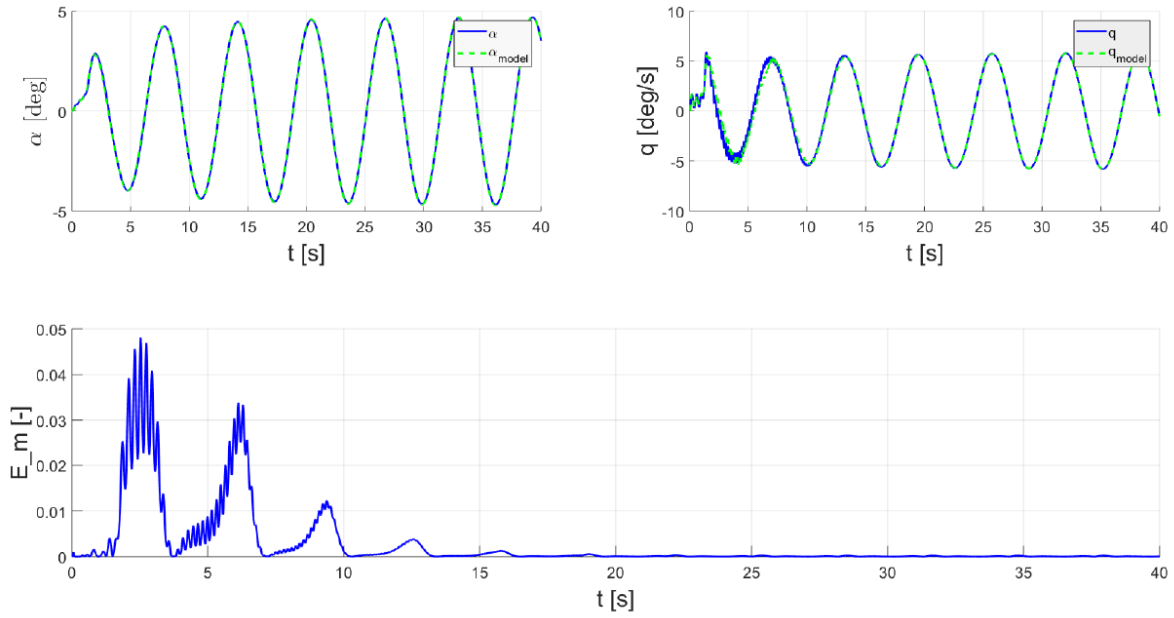


Figure (5.16) Model identification for nonlinear missile model with HDP controller for angle of attack identification (top left), pitch rate identification (top right) and over all model error (bottom)

Figure 5.17 shows the weights for the actor, critic, and model networks output layer and hidden layers over time.

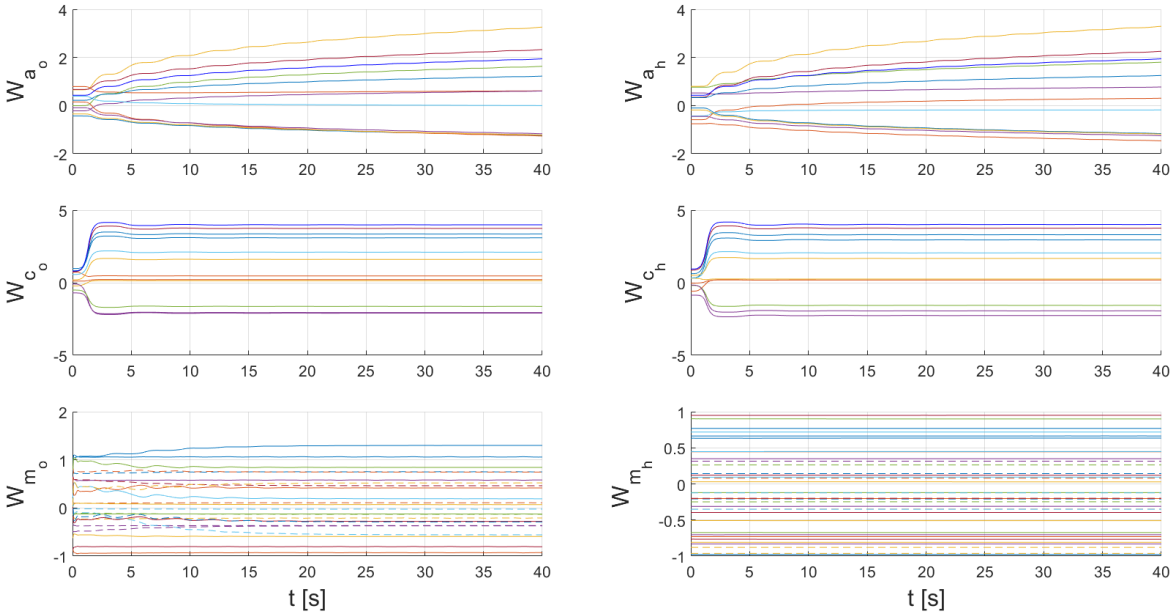


Figure (5.17) Weight updates with HDP controller for missile model with angle of attack identification for actor output layer weights (top left), actor hidden layer weights (top right), critic output layer weights (middle left), critic hidden layer weights (middle right), model output layer weights (bottom left) and model hidden layer weights (bottom right)

The actor networks show consistent changes over online learning while the critic network converges after the first 10 seconds of learning. From the graph, it is evident that the model weights changes become much less after 15 seconds, and some of the weight changes are only within the first time steps. These weights are related to the angle of attack state inputs, as the model starts to follow the angle of attack response much quicker, and in the first time steps of the learning. The results in this section show that the HDP controller can follow the reference signal carefully only after 20 seconds, and the tracking improves even further after this. However, as demonstrated by literature, this performance can be enhanced if the PE condition is applied so that the controller will have more exploration in the very first few seconds of online learning. Therefore, in the next section, PE condition is added to the HDP implementation, while some adjustments to the parameters are made.

5.3.6. Results for heuristic dynamic programming for nonlinear missile model with PE

The results for the HDP implementation with persistent excitation and zero initial states are shown in Figure 5.18. The noise is added only to the first 5 seconds of online learning. As shown by the figure, the angle of attack tracking is feasible with the HDP implementation with the added excitation. Looking at the angle of attack tracking graph indicates that the HDP algorithm controls the angle of attack to track very closely the reference signal, which is a sine function with an amplitude of 5 degrees. The tracking error is presented in the cost function during the tracking task, as it is the direct function of the tracking error.

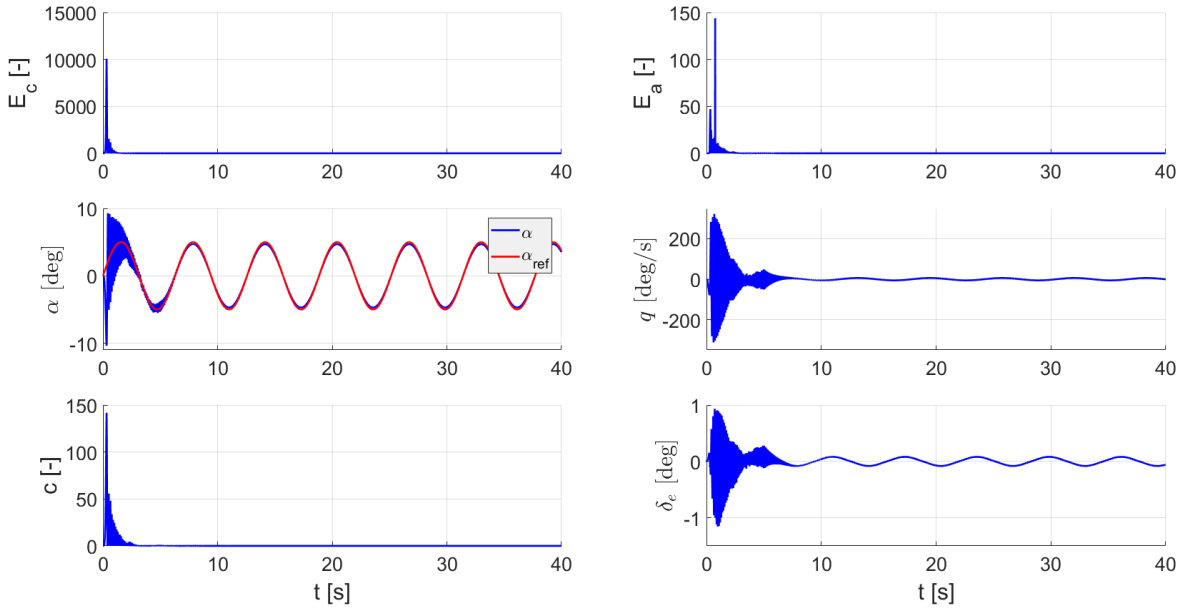


Figure (5.18) Results of nonlinear missile model with HDP controller with added PE for critic error (top left), actor error (top right), angle of attack tracking (middle left), pitch rate response (middle right), cost values (bottom left) and elevator deflection (bottom right)

When compared to the case without PE condition, it can be seen that with the PE condition met, the HDP controller follows the reference signal more precisely, and the error drop to zero faster. To show the results after the PE is removed, the angle of attack tracking, pitch rate response, and elevator deflections control inputs are shown separately by Figure 5.19. The angle of attack response rejects the disturbance after 5 seconds, while for the pitch rate and elevator deflection, the disturbances are rejected only after 10 seconds. When compared to the HDP controller with no PE, the HDP controller, in this case, reach the (near)optimal policy sooner, and few to no change can be seen in the policy after 10 seconds. Looking at Figure 5.15, the policy continually changes during the online learning towards the (near)optimal policy.

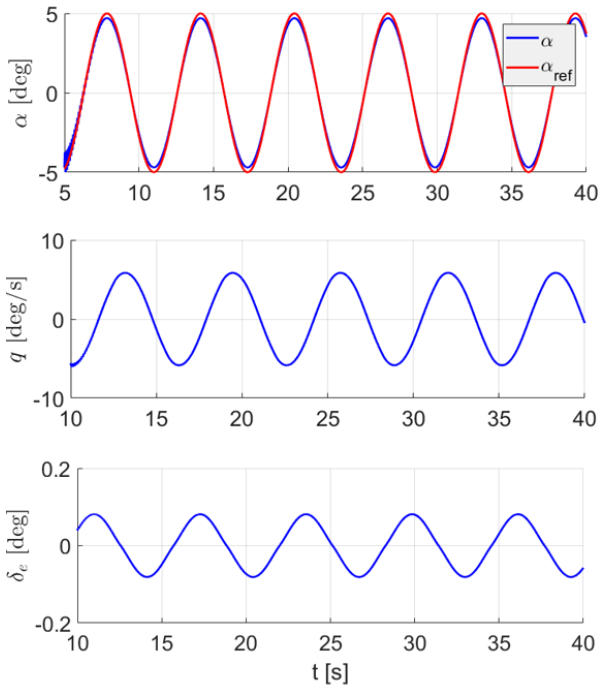


Figure (5.19) Zoomed in results of nonlinear missile model with HDP controller with added PE for critic error (top left), actor error (top right), angle of attack tracking (middle left), pitch rate response (middle right), cost values (bottom left) and elevator deflection (bottom right)

Looking at the model learning graphs in Figure 5.20, the excited HDP controller identify the global model as precise as the HDP controller with no PE. However, the disturbance can be seen with significant variations in the recognized model in the first 5 seconds. These variations die out gradually after the noise is removed. This is not the case for the previous case controller, as one can see that the model is identified in the very first time steps of the learning without large spikes in the angle of attack and pitch rates. These large variations can be dangerous in reality as they can lead to the failure of the system and loss of control. Same as the behavioral plots, The model error goes very big in the first few seconds. These large spikes cannot be seen in the HDP controller without PE, as the model error stays within a boundary of less than 0.05, while in the present case, the error becomes in the order of 10^4. Looking at the angle of attack and pitch rate identification, the large spikes can also be seen in these graphs, especially for the pitch rate identification. However, when the state errors are looked at separately, the error for both states seems to die out to zero. To further explore the experiment, the model learning graphs are zoomed in for angle of attack after 5 seconds and for the pitch rate after 10 seconds, as shown by Figure 5.21. Looking at the graph, it can be seen that the model is identified by the controller precisely after 5 seconds for the angle of attack and after 10 seconds for the pitch rate.

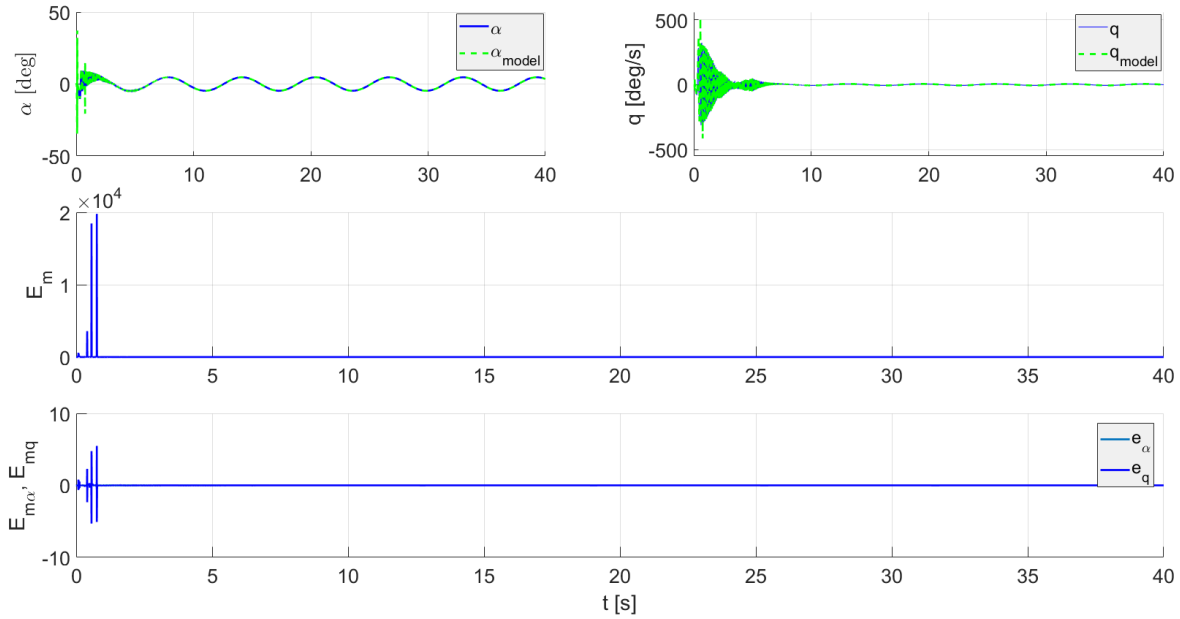


Figure (5.20) Results of nonlinear missile model with HDP controller for critic error (top left), actor error (top right), angle of attack tracking (middle left), pitch rate response (middle right), cost values (bottom left) and elevator deflection (bottom right)

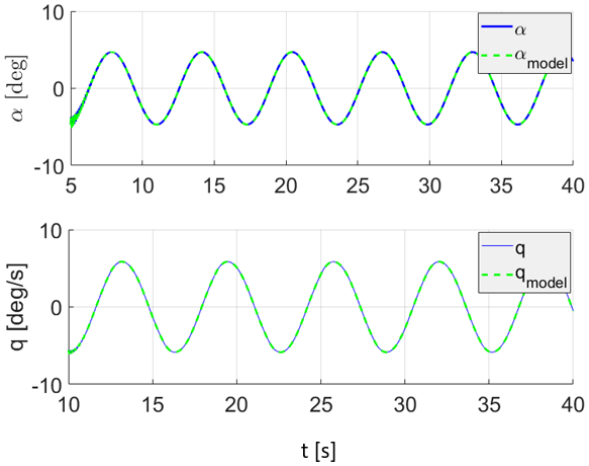


Figure (5.21) Zoomed in results of nonlinear model with HDP controller for critic error (top left), actor error (top right), angle of attack tracking (middle left), pitch rate response (middle right), cost values (bottom left) and elevator deflection (bottom right)

The model error for each state is zoomed in by Figure 5.22 after first 2 seconds. As one can see, the model error goes to zero after 3 seconds of online learning.

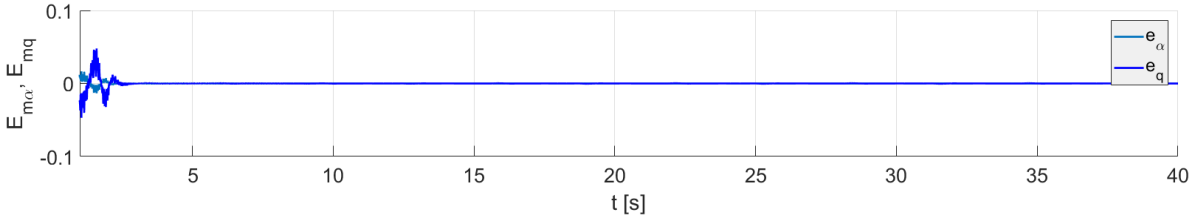


Figure (5.22) Results of nonlinear missile model with HDP controller for critic error (top left), actor error (top right), angle of attack tracking (middle left), pitch rate response (middle right), cost values (bottom left) and elevator deflection (bottom right)

The weights progress are shown by Figure 5.23. The graph shows that the actor weights converge in

less than 3 seconds, while for the critic network convergence happens even sooner. It was shown that for the HDP controller with no PE condition met, the actor weights were slowly changing during the online learning. Symmetry can be seen in the critic hidden and output layer weights, which can mean that some of the layer outputs might cancel each other out. That means that fewer critic neurons might also be acceptable. The initial variations in the model network are also reflected in the output layer weights of the model network and less in the hidden layer.

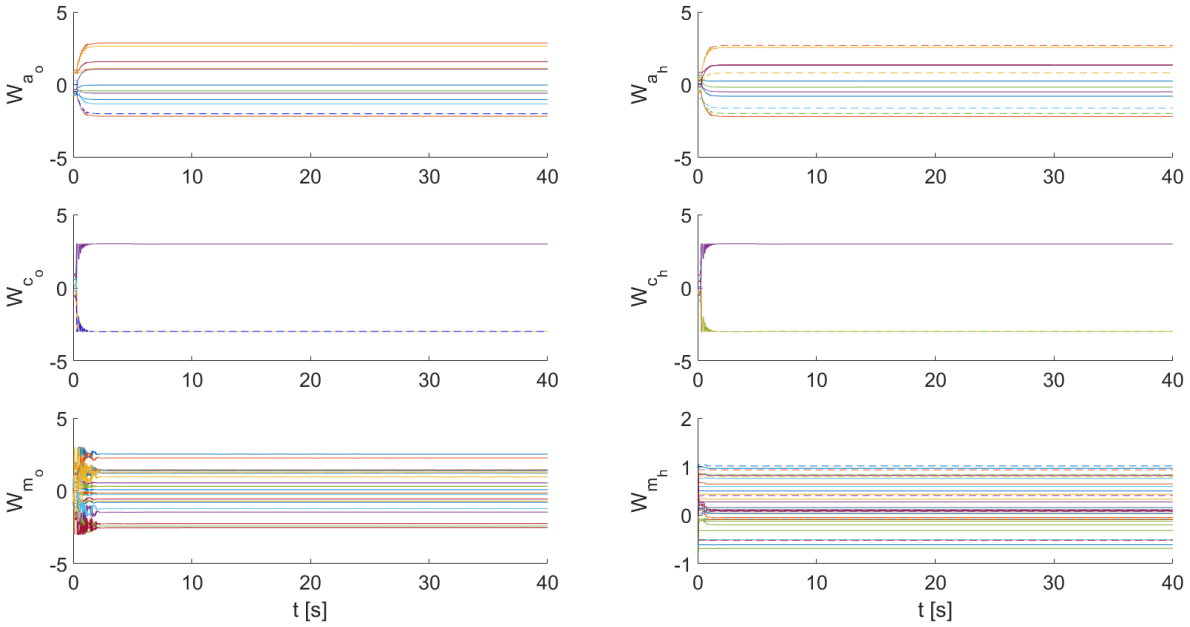


Figure (5.23) Results of nonlinear missile model with HDP controller for critic error (top left), actor error (top right), angle of attack tracking (middle left), pitch rate response (middle right), cost values (bottom left) and elevator deflection (bottom right)

To check the stability of the current algorithm, as stability is an essential aspect in online learning, a measure needs to be defined. The learning stability can be defined as the number of times the controller converges to a stable solution when it is given random e.g., weight initialization. The measure is referred to as the success rate, and it is shown by percentage. For this experiment, the HDP controller with PE runs for 100 times with random initialization, and the results for the cumulative cost, cumulative actor error, and cumulative critic error in addition to cumulative model error are used to analyzing the stability of the controller. It should be noted that for hence of a fair comparison, the same PE condition is used for all the runs. As described by Equation 5.16, the cost per time step increased exponentially with the error. Therefore, with larger errors, the weighing is larger compared to small errors. As an example, based on the current definition of the cost function, the minimum value for the cost function is 0 while the maximum value can go in order of 10^3 for deviations of 1 deg from the reference angle of attack and therefore, the cumulative value will be in the order of 10^8. The run presented earlier in this section as a successful run has a cumulative cost of 6.9314×10^4, and cumulative actor, critic, and model errors of 1.9385×10^4, 1.8793×10^6 and 2.5742×10^5.

To describe the success rate definition, a threshold for the failure need to be defined. Based on the elaboration above, failure has happened if the critic neural network weights 'exploded' resulting in a NaN output for the critic forward pass. This would then result in the same NaN value for the rest of the network outputs. With such a definition, the success rate for the HDP controller for 100 runs is 58% . Figure 5.24 shows the histogram for the cumulative cost, cumulative actor, critic, and model errors to give a better presentation of the HDP controller. The graph shows all 100 run results with 11 bins in a range of $[0.4634, 16.9591]$, $[0.0632, 3.4707]$, $[0.0470, 4.9136]$ and $[0, 10.7892]$ for cumulative cost and cumulative actor, critic and model errors with the collective values scaled with 10^5 for cost and actor error, 10^7 for critic error and 10^8 for the model error. The graph shows that the HDP controller can converge with a reasonable likelihood as the first bin has the highest occurrences. As the cost function values and network errors become more significant, the graph shows fewer occurrences.

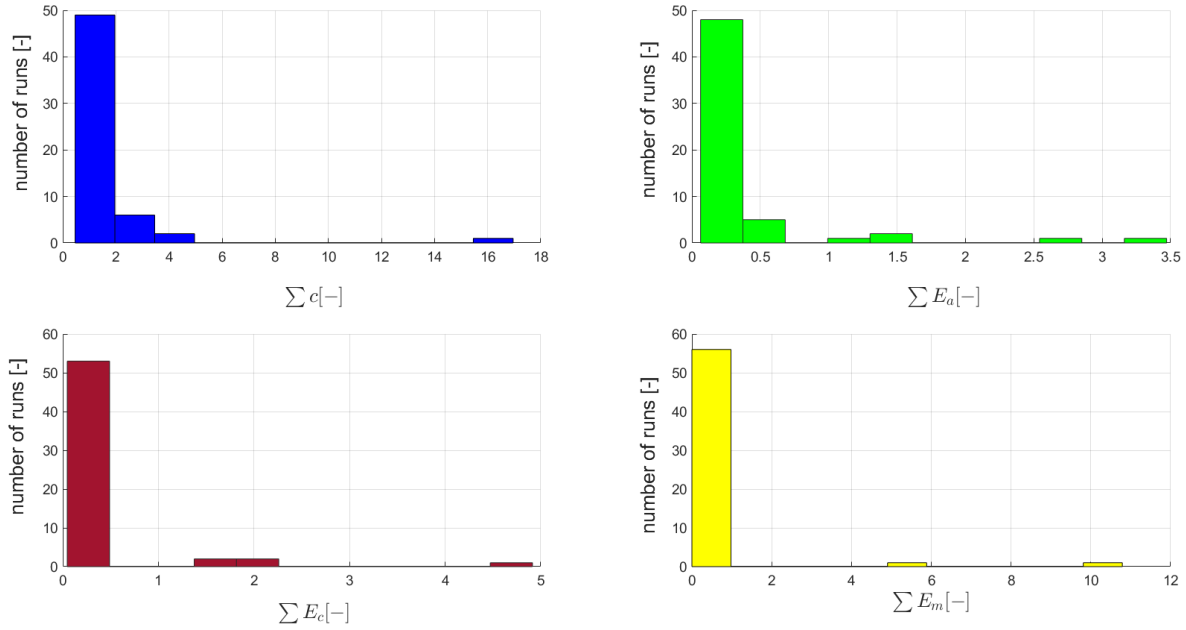


Figure (5.24) Histogram of cumulative cost (top left), cumulative actor error (top right), cumulative critic error (bottom left) and cumulative model error (bottom right) for 100 runs

5.4. Conclusion

This chapter included preliminary experiments with discrete and continuous/approximate reinforcement learning control approaches to design adaptive controllers that require no prior knowledge of the system dynamics. To show the control performance of the discrete and approximate RL controllers, this chapter applied discrete Q-learning and continuous method of heuristic dynamic programming on two highly nonlinear control problems of rotatory pendulum swing-up and stabilization and missile tracking problem. The first control problem includes two control objectives of first swinging-up the pendulum and then stabilize it in the upright position (e.g., if the pendulum starts from the bottom position). Therefore, using the RL approach for this control problem shows the advantage of this method in achieving challenging control objectives without prior knowledge of the system. To demonstrate the inherent ability of RL controllers in control allocation for an over-actuated problem, the Q-learning approach is used for the over-actuated pendulum problem with two available action inputs of torque and added force. To see the control performance of the continuous methods, the widely used HDP method with a general nonlinear cost function is used for the tracking problem of an unknown nonlinear missile model.

The results show that the discrete Q-learning can achieve the control objective in both cases were only one control input is available for the controller and in case of the over-actuated control problem. While with more classical control approaches, the fixed input-output structures require the prior adjustments to the explicit definitions, for the RL controller, the over-actuated problem means an increase in the action space. The reward function used in the RL-setup only considers the proximity to zero for the pendulum angle as the goal in both case studies. In both cases, a similar behavior was shown in terms of learning behavior and the value-function, although in case of the under-actuated problem, the control objective is achieved sooner, and the pendulum reaches the control objective faster as a more difficult problem is given to the controller. The conceived discrete RL-based control approach method proved to be able to tackle the control allocation problem and distribute the control inputs utilization with the expense of an increase in dimensionality and computing time. Furthermore, it was shown that the discrete approach converges to a (near) optimal policy of a bang-bang control. In reality, this control policy is not preferable as the control effector will be used to their limits. Therefore, the approximate RL approach which is shown to tackle with the ”curse-of-dimentionality” and consistency of the control policy was used for the nonlinear problem of missile control.

The HDP controller was applied to the missile tracking problem in two cases, first with no persistent excitation and the second case where the PE condition was met. The HDP results on the missile tracking problem demonstrate that the HDP method can track the reference sinusoidal signal without prior offline training and with no in advance knowledge on the system dynamics. Additionally, it was shown that the performance of the tracking improved with the added PE condition as the controller follow the reference signal precisely from the very beginning of learning with the expense of added noise in the first few seconds due to the added exploration. Also, in this case, the controller converges to the (near) optimal policy faster, and the policy remains unchanged for the rest of the online learning, while with no PE added, the policy keeps changing towards the (near) optimal policy. Therefore, with PE condition met, the control policy shows consistency after the few seconds of online learning. The angle of attack and pitch rate models identified precisely after the very few seconds, and the controller can reject the disturbances fast.

To conclude, this chapter showed abilities of the RL controller in solving highly nonlinear and over-actuated control problems with no prior knowledge about the system by implementing discrete Q-learning and then demonstrated how an approximate RL method of HDP can solve a similar nonlinear problem online while overcoming some deficiencies of discrete RL problems. Therefore, this chapter generalized the use and applications of discrete and continuous RL methods.

III

Additional Results and Final Remarks

6

Sensitivity Analysis for ICE Aircraft HDP Controller

To investigate the effect of the learning parameters on the controller and its computational performance, a sensitivity analysis is performed for the ICE aircraft HDP controller. Also, the robustness and adaptability of the proposed attitude RL controller are investigated in this chapter. The research case used for this purpose is the angle of attack tracking with the possibility for the controller to use pitch flaps and elevons effectors. The properties of the HDP controller and the ICE aircraft are the same as what is shown in the scientific paper in this report (RC2). In the following, the sensitivity analysis conditions are described, followed by the results. In the final section, the robustness and adaptability of the designed controller are discussed, and their results are presented.

6.1. Sensitivity Analysis Conditions

In the following, the conditions are shown for sensitivity analysis on the number of neurons used for the critic, actor, and model networks. It should be noted that for each case, the same amount of neurons are used for each network. The table below shows these conditions for sensitivity analysis. The change in the number of neurons directly affects the complexity of the network. It usually is recommended to use the least number of neurons to achieve the desired approximation power. Therefore, choosing the right number of neurons means a trade-off between computational complexity and model performance typically. Few number of neurons can decrease the capacity of the network in learning the underlying pattern. A big number of neurons can result in a considerable learning duration. Therefore, as computational effort is crucial for an online controller, these parameters are investigated here.

Table (6.1) Sensitivity analysis conditions for actor, critic and model networks number of neurons

Case	Neurons No.	Effectors
SA1	6	PF, Elevon
SA2	12	PF, Elevon
SA3	25	PF, Elevon
SA4	36	PF, Elevon

The 25 number of neurons were used for developing results in the scientific paper and is the baseline for generating results for attitude and altitude controllers. Therefore, 6, 12, and 36 numbers of neurons are chosen to show the effect of this parameter in the sensitivity analysis.

6.2. Learning Rates

Another parameter that has an essential role in the performance of an RL controller is the learning parameter. Learning rates control the adaptability of an RL controller. This parameter is the most crucial in the convergence and stability of the controller and also in its convergence time. Therefore, sensitivity analysis in this section is performed for learning rates for each network, so the actor, critic, and model networks. The learning rates are chosen by changing their order of magnitude.

Table (6.2) Sensitivity analysis conditions for critic, actor and model learning rates

Case	learning rate	Case	learning rate	Case	learning rate	Effectors
$SA1_{\alpha_c}$	0.1	$SA1_{\alpha_a}$	10	$SA1_{\alpha_m}$	0.01	PF, Elevon
$SA2_{\alpha_c}$	1	$SA2_{\alpha_a}$	500	$SA2_{\alpha_m}$	0.15	PF, Elevon
$SA3_{\alpha_c}$	10	$SA3_{\alpha_a}$	1000	$SA3_{\alpha_m}$	10	PF, Elevon

The baseline is the learning rates that are optimized for the results shown for the ICE aircraft, so the actor learning rate of 500, the critic learning rate of 1, and the model learning rate of 0.15. If the learning rate is too big, it usually means that the model will learn faster with the cost of network weights converging to a sub-optimal solution. Small learning rates can increase the chance of achieving optimal weights (even globally) but can increase the training time.

6.3. Weight Initialization

The random weight initialization is another aspect of function approximation methods that is important in the learning process. Neural network weights normally are initialized with small random numbers. The careful initialization of the learning process can affect the learning speed and convergence of the algorithm. Therefore, in this section, the effect of the weight initialization is investigated by defining three conditions. These conditions are differ based on the boundary defined for the weights, and different techniques or distributions are not considered here.

Table (6.3) Sensitivity analysis conditions for weight initialization

Case	random weight initialization range
$SA1_{W_0}$	[-0.1, 0.1]
$SA2_{W_0}$	[-1.0, 1.0]
$SA3_{W_0}$	[10, 10]

The neural networks designed for the three networks of the HDP controller for the ICE aircraft do not include a bias term to reduce the complexity in updating the weights of the network, and they are initialized using symmetric random number generator within [-1,1].

6.4. Sensitivity Analysis Results and Discussion

For this experiment, one of the best results obtained for the angle of attack tracking, which was obtained using two effectors of pitch flaps and elevons are considered for sensitivity analysis basis. In this section, the results for the sensitivity analysis described above are given both qualitatively by presenting graphs and quantitatively by table presentation. The Root Mean Square Error (RMSE) for the angle of attack tracking, the run time duration, and the Mean Control Effort (MCE) for the control effectors are used for the comparison. The MCE is defined as:

$$MCE = \frac{\sum \delta_{eff}/\delta_{eff_{max}}}{N} \tag{6.1}$$

where δ_{eff} is the effector deflection, $\delta_{eff_{max}}$ is the maximum deflection for the effector and N is the number of sample for each experiment condition.

6.4.1. Number of neurons

The results for ICE aircraft longitudinal attitude control using pitch flaps and elevons for a different number of neurons for the critic, actor, and model networks are presented in this section. This experiment runs in MATLAB on a PC of i5-6500 CPU and 8 GB of RAM. The controllers are compared in terms of the Root Mean Square (RMS) of the tracking error, the simulation run time, and the control surfaces Mean Control Effort (MCE). The quantitative results of the sensitivity analysis are presented by Table 6.4.

Table (6.4) The tracking performance for each benchmark condition

Case	RMS_α [deg]	Run Time [seconds]	PF MCE [-]	ELE MCE [-]
SA1	1.5272	36.5558	0.0565	0.0975
SA2	1.4679	38.9598	0.0443	0.0691
SA3	0.2251	36.0391	0.0703	0.0797
SA4	1.9314	39.8050	0.1654	0.1576

The results in Table 6.4 shows that the number of neurons has an impact on the attitude tracking performance while there is not a big difference in the RMS of the tracking errors for the first two and the last conditions. However, it can be seen that the performance is significantly better for the cases of SA3 and slightly better for SA2 when compared to cases of SA1 and SA4. Therefore, the choice of 25 neurons for the experiment runs of ICE aircraft seems reasonable, base on the RMS of the angle of attack tracking error.

Another criteria to discuss is the run time. One would expect the run times to increase with the increase in the number of neurons. However, the run time values show a small difference from each other, with the last case run time increase of 3 seconds for this case compared to the first case. SA1 and SA3 have the shortest run times with similar values, while the number of neurons in the latter case is almost twice the first case. As different policies can be obtained to provide an optimal solution to the tracking problem, the control efforts are compared here. As one can see, the control effort in the last case is more for both the PF and ELE effectors, while for both effectors least control effector is obtained by case SA2. It means that different (near) optimal solutions are obtained, and no direct conclusion can be drawn from the relation of control effort and the number of neurons.

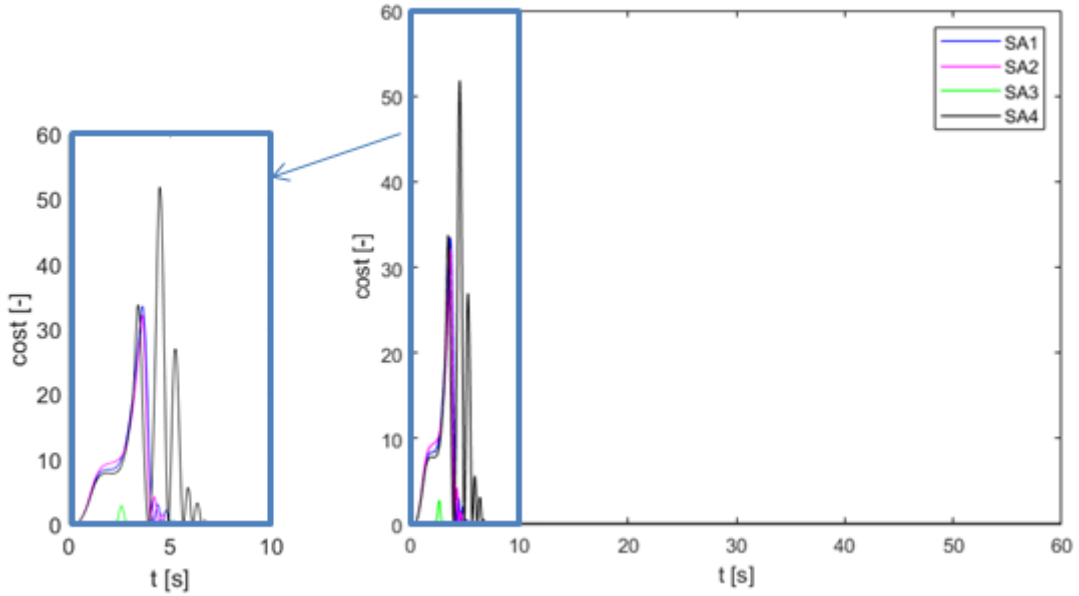


Figure (6.1) Cost values for different number of neurons in attitude control of ICE aircraft using pitch flaps and elevons as inputs

Another way to see the performance of the controllers is by showing cost values in the learning process. In Figure 6.1 the cost values are shown for each sensitivity analysis. It can be seen that for the SA3 case, the cost values are significantly smaller compared to the other 3 cases. And bigger cost values are obtained by the last case. Another observation is that the cost values converge to zeros much faster for the first three conditions, with the SA1 and SA2 conditions showing a very close trend for cost changes. It should be noted that the cost coefficient has a large value for the angle of attack tracking problem (order of 10^3). Therefore, the large scales of the errors are obtained based on this large value.

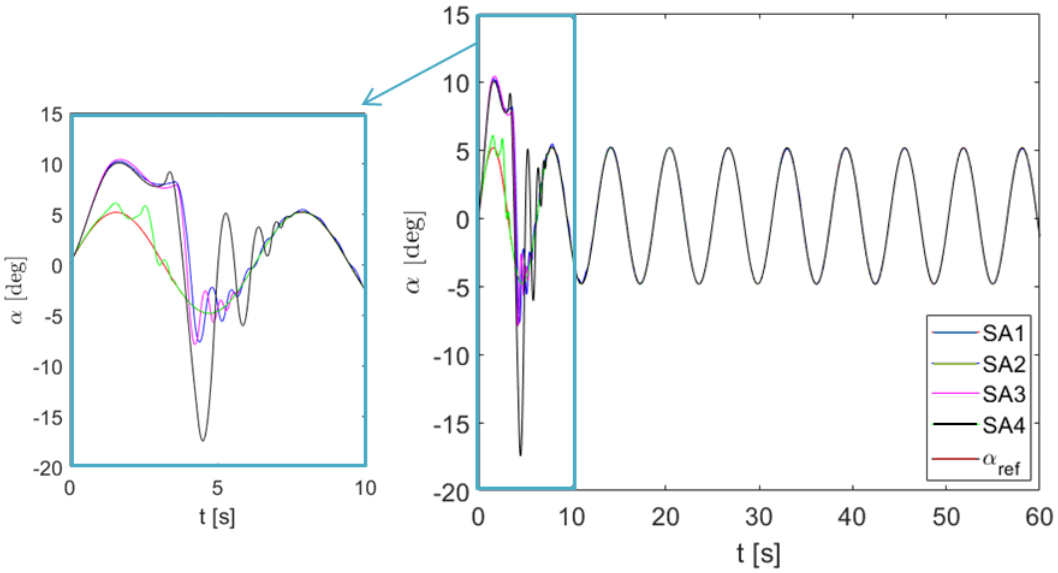


Figure (6.2) Tracking performance for different number of neurons in attitude control of ICE aircraft using pitch flaps and elevons as inputs

The Figure 6.2 shows the tracking signal and the angle of attack response. As expected the, a best significant performance is achieved by case SA3 and the worse performance by the SA4. Therefore, the increase in the number of neurons does not necessarily increase tracking performance.

The pitch flaps and elevons deflections for all the cases are presented separately by Figure 6.3 and Figure 6.5, respectively. The figures for both effectors show a very high frequency change in the deflections between 3 to 5 seconds for the last sensitivity case with 36 number of neurons. The high frequency variations also happen for the other first few seconds of the controllers with less number of neurons as can be seen by the additional Figure 6.4. In this figure, the plot corresponding to SA4 is omitted for better visualization of the other graphs. As the figure shows in terms of avoiding pitch flap saturation limits and frequency of the changes, the first case with 6 number of neurons shows a smoother and better practical policy for pitch flaps. It can be seen that the controllers obtain different control policies for the pitch flaps by looking at the overall response and especially the last 10 seconds. However, it can be seen that the SA2 and SA3 cases adopt a similar policy, but with little difference in scales while for the first case the pitch flap deflection is in the opposite direction as the other cases and for case SA4, the pitch flap deflections become zero after about 8 seconds. Next, the control policy for the elevon effectors for different research cases is shown by Figure 6.5. From the figure, it can be observed that the elevons control policy has nearly the same trend with small differences in the amplitudes for all the cases. As it was observed that the case SA1 obtain a policy which is less aggressive in the beginning, however, when looking at both graphs of Figure 6.3 and Figure 6.5, one can see that in this case the effectors are deflecting antagonistically and producing excess drag. This is while for the two cases of SA2 and SA3, the effectors are utilized in a contributing way to increase the longitudinal control power. For the last case of SA4, the controller is not using the pitch flaps and only use elevons with the bigger amplitude. And the amplitude of using elevons becomes the biggest for the case SA1 where the effectors are deflecting in opposite directions. Therefore, it can be seen

that the number of neurons affects the policy that the controller adopt and converge to.

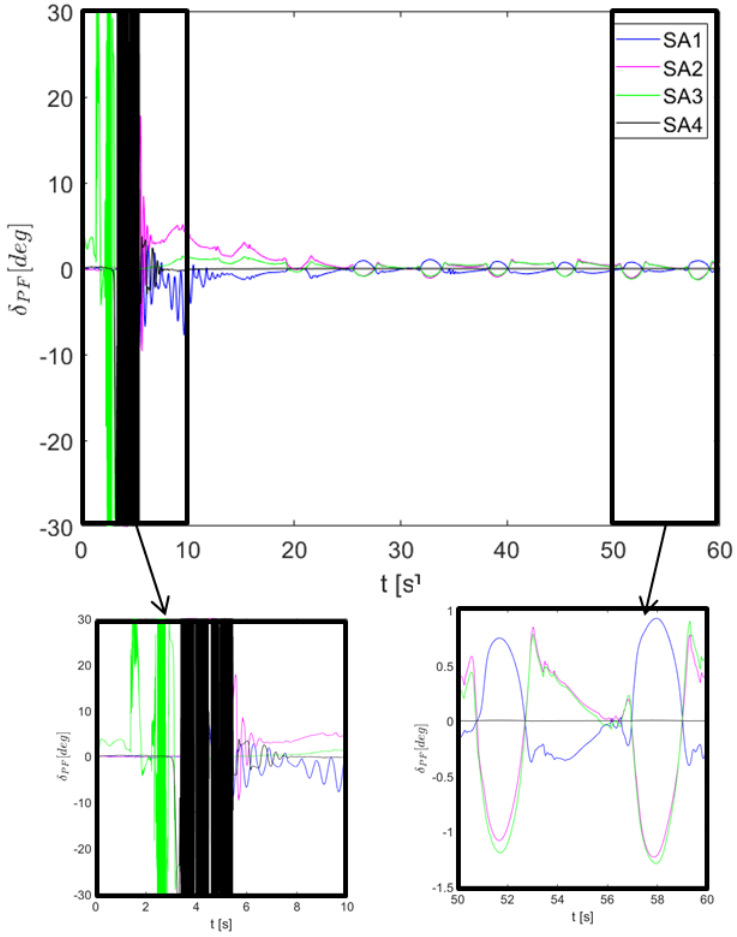


Figure (6.3) Pitch flap control input response for different number of neurons in attitude control of ICE aircraft using pitch flaps and elevons as inputs

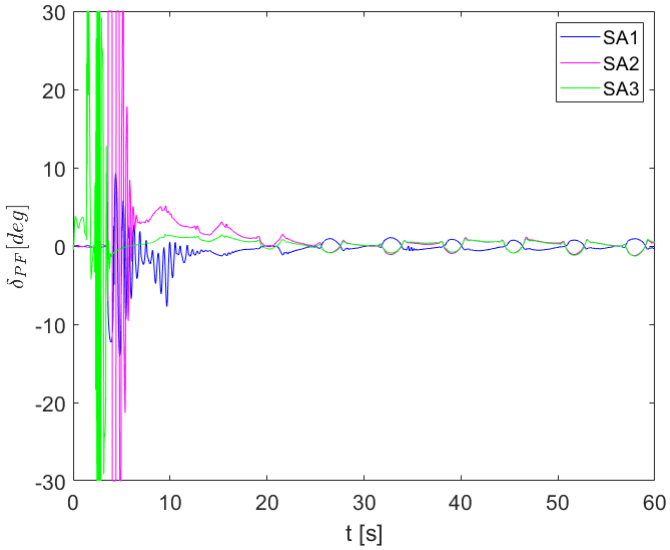


Figure (6.4) Pitch flap control input response for different number of neurons in attitude control of ICE aircraft using pitch flaps and elevons as inputs (SA4 omitted)

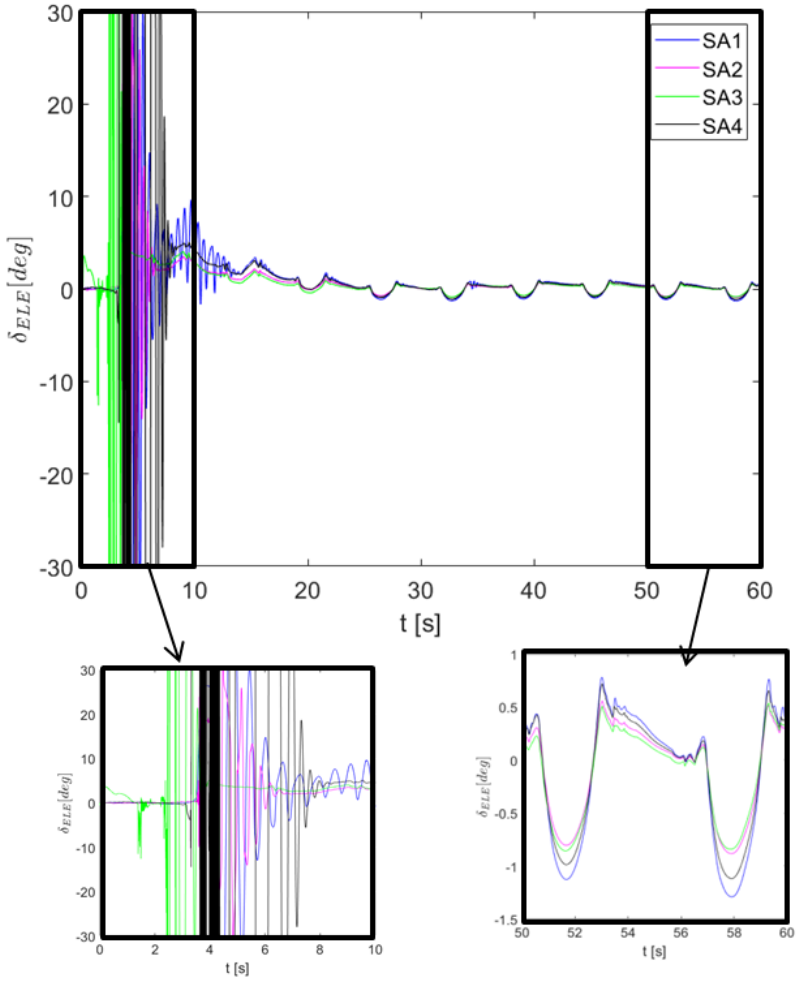


Figure (6.5) Elevon control input response for different number of neurons in attitude control of ICE aircraft using pitch flaps and elevons as inputs

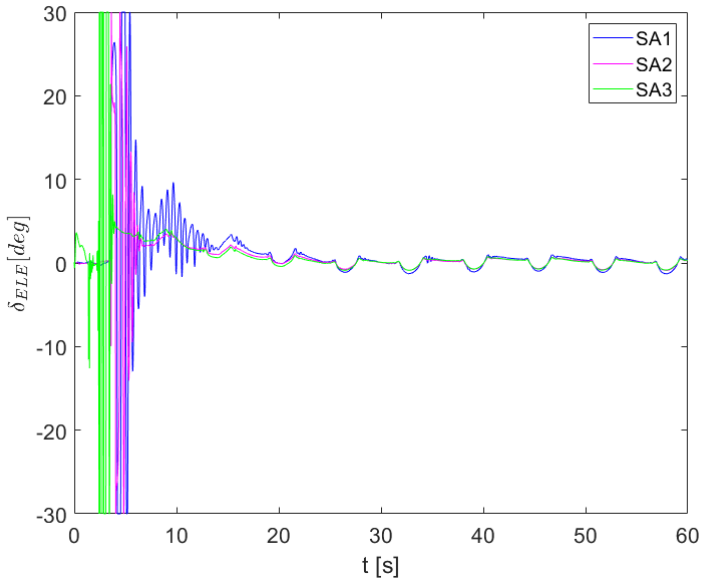


Figure (6.6) Elevon control input response for different number of neurons in attitude control of ICE aircraft using pitch flaps and elevons as inputs (SA4 omitted)

6.4.2. learning rates

Next, the learning rates are compared quantitatively for each neural network structure. The comparison is performed for the success ratio of the controllers. Table 6.5 shows the results for the sensitivity analysis. While the learning rates are changed by order of magnitude, it can be seen that for the critic network, the stability of the controller is better for the case of $SA2_{\alpha_c}$ and the success ratio is more significant for the case $SA1_{\alpha_c}$ compared to $SA3_{\alpha_c}$. Therefore, a general conclusion can not be derived for the critic learning rate, as decreasing or increasing the critic learning rate does not necessarily improve the stability of the controller, while the controller shows more stability for critic learning rate of order 0.

For the actor network, again the case SAw_{α_a} shows the most significant success ratio while the case $SA3_{\alpha_c}$ with larger learning rate shows a better performance than the $SA1_{\alpha_c}$ with lower rates. Comparing the effects of learning rates between critic and actor it can be observed that increasing learning rates result in worse stability while decreasing actor learning rate would deteriorate the stability more for the three cases. In this case, better performance is obtained by the large actor learning rate of order 2.

Looking at the model learning rates, the $SA2_{\alpha_m}$ and $SA3_{\alpha_m}$ show close success ratios but with both of them better than case $SA1_{\alpha_a}$, where the success ratio is much smaller for this case. While the model learning rate order has increased by 2, there is not a much effect on the stability of the controller, while making the model learning very small make the stability of the controller worse.

Table (6.5) The success ratio for each benchmark condition

Case	Success Ratio [%]	Case	Success Ratio [%]	Case	Success Ratio [%]
$SA1_{\alpha_c}$	31	$SA1_{\alpha_a}$	26	$SA1_{\alpha_m}$	24
$SA2_{\alpha_c}$	38	$SA2_{\alpha_a}$	38	$SA2_{\alpha_m}$	38
$SA3_{\alpha_c}$	21	$SA3_{\alpha_a}$	30	$SA3_{\alpha_m}$	37

6.4.3. weight initialization

In the last attempt for the sensitivity analysis on the designed attitude controller for the ICE aircraft, the network's weight initialization range are selected for comparison. The symmetric random number generator is used to select the initial weights of each of the three networks. One hundred runs are performed, and the success ratios are shown in Table 6.6 to have a measure for the stability of the designed HDP controller. As one can see, the range of the weight initialization has a significant impact on the success ratio and convergence of the controller for the ICE aircraft. As with the minimal weights, the controller only become successful in tracking the reference signal once, while with a more significant range for weights, the success ratio becomes much larger. Therefore, it is evident that this parameter makes a fundamental change in the convergence and thus, the stability of the controller.

Table (6.6) The success ratio for each benchmark condition

Case	Success Ratio [%]
$SA1_{W_0}$	1
$SA2_{W_0}$	38
$SA3_{W_0}$	56

6.5. Robustness and Adaptability

To see the robustness and adaptability of the designed attitude controller for the ICE aircraft, two cases are presented here. The simulation time increases for both cases of robustness check and adaptability checks, to 120 seconds. In the first case, the weights of the converged and online trained RL controller will be frozen after 60 seconds, and a different trajectory signal will be given to the controller to follow with no additional online training performed. In the second case, however, online learning will continue

after 60 seconds while the controller will be given the new trajectory. The results are shown in the following sections.

6.5.1. Robustness of the HDP controller

To investigate the robustness of the HDP controller, the attitude control problem with two available control surfaces of pitch flaps and elevons are also considered for this experiment. Four reference signals are given to the controller after the 60 seconds of online learning. At this instance, the controller will not be trained any further. The first signal is a constant angle of attack to follow. This constant value equals the mean of the original sinusoidal signal. Next, the amplitude of the reference sinusoidal signal (from the first 60 seconds) is multiplied by 3. For the third case, a cosine signal is used with a bigger amplitude than the original tracking signal and a smaller frequency. For the last case, the signal amplitude becomes bigger while the frequency of the signal increases. Therefore, with these research cases, the robustness of the controller in following a constant signal and a signal with different nature and with different frequencies will be investigated. In table Table 6.7 these conditions are shown.

Table (6.7) The reference signals used for checking the robustness of the HDP controller

Case	Reference Signal (deg)
R1	$\alpha_{ref} = 0.1503$
R2	$\alpha_{ref} = 15sin(t)$
R3	$\alpha_{ref} = 15cos(\frac{t}{\pi})$
R4	$\alpha_{ref} = 15sin(\frac{\pi}{2}t)$

Figure 6.7 shows the results for the four different cases. It should be noted that the plots show the online trained controller behaviour after it is trained to perform the first reference signal and therefore, the initial online training is omitted, and it is assumed that the controller has already learned the reference signal. To give a better presentation for the tracking performance in the instant of signal changing, the graph is zoomed for these periods, which is shown by Figure 6.8. Looking at the two plots, it can be seen that the HDP controller is successful in following all the presented different signals rapidly after the signal change. As the controller was trained with the tracking error, it can follow the signal precisely even without any further online learning. The performance shows some deterioration for the R3 case where both the nature of the reference signal and the frequency of the signal has changed as the angle of attack response increase to 20 degrees (still within the range of the acceptable response for the angle of attacks). However, in this case, also, the controller can follow the reference signal less than a second.

Figure 6.8 with the zoomed-in responses, shows that the controller can follow the new reference signal less than half a second for the R1, R2 and R4 cases while for R3 case the good tracking is achieved before 1 second. The graphs show the power of the designed controller and its robustness to changes in the reference signal. However, to make sure that these manoeuvres are possible with the current ICE configuration, the control effectors behaviour should be checked. Therefore, next the policy obtained by the controller after the change in the tracking signal is investigated by plotting the effectors' deflection for this period.

To see the change in the control policy with the new signals given to the online trained controller, the control effectors behaviours are also shown for the four different cases by Figure 6.9 with the zoomed-in results for the duration when the reference signal is changed by Figure 6.10.

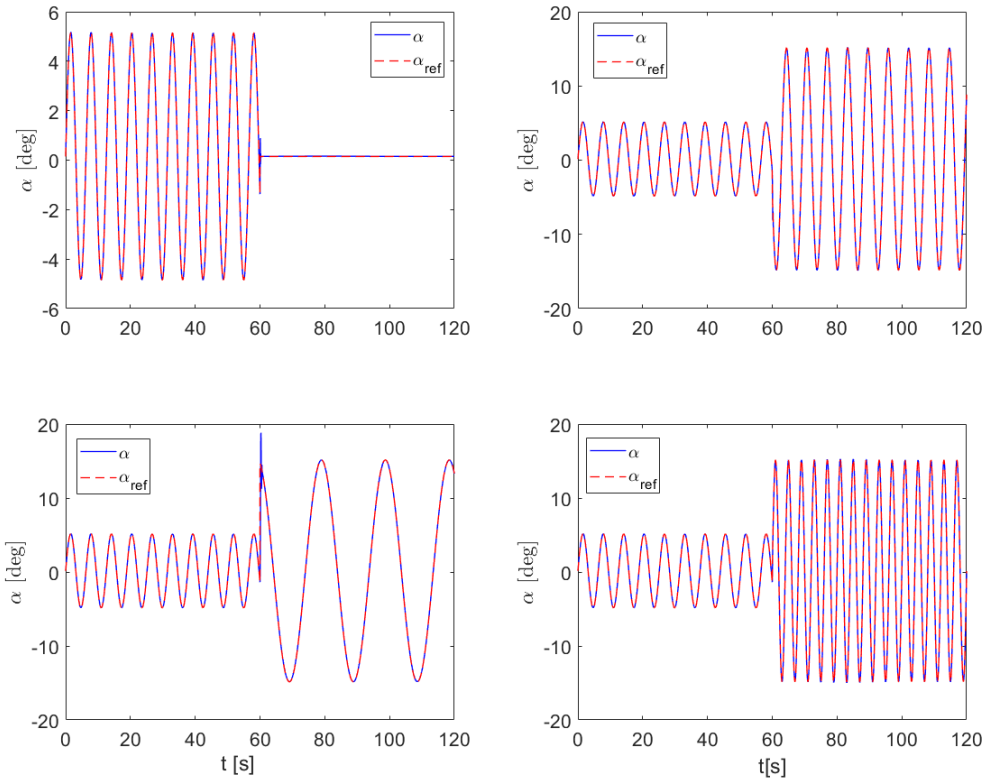


Figure (6.7) The results for checking the robustness of the HDP angle of attack controller for different angle of attack reference signals for R1 case on top left, R2 case top right, R3 case bottom left and RC4 bottom right

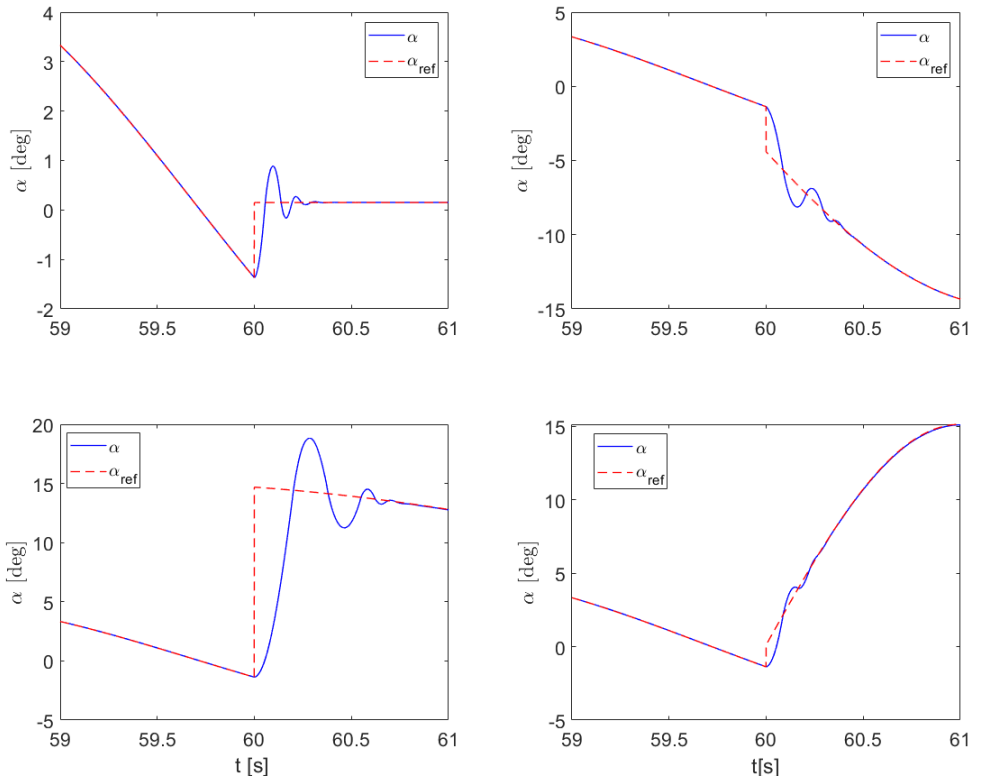


Figure (6.8) Zoomed in results for checking the robustness of the HDP angle of attack controller for different angle of attack reference signals for R1 case on top left, R2 case top right, R3 case bottom left and RC4 bottom right

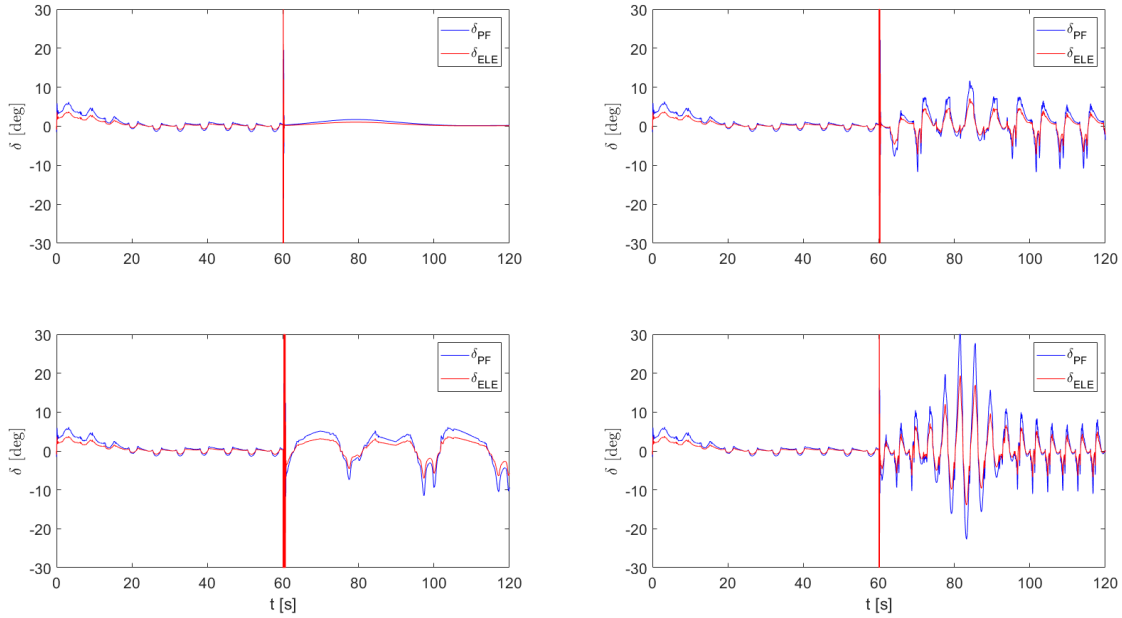


Figure (6.9) The control policies for checking the robustness of the HDP angle of attack controller for different angle of attack reference signals for R1 case on top left, R2 case top right, R3 case bottom left and RC4 bottom right

It can be seen that the controller adopt a new policy with the change in the reference signal and make changes to this policy for the rest of the online control. It can also be observed that with the difference in the reference signal, the controllers are pushed to their limits and quickly adopt a different policy for all cases. For the case of R3, the controller take a few time steps more to change for the new policy as could have been expected as the signal has changed in nature for this case while the tracking signal itself is more complicated (e.g., compared to case R1). It can be seen that the control policy obtained use two effectors in the same direction and therefore, the effectors contribute to the tracking control. For case R4, where the signal frequency has increased, it can be seen that the effectors' deflections are also changing with a high rate which can be undesirable for real-world cases.

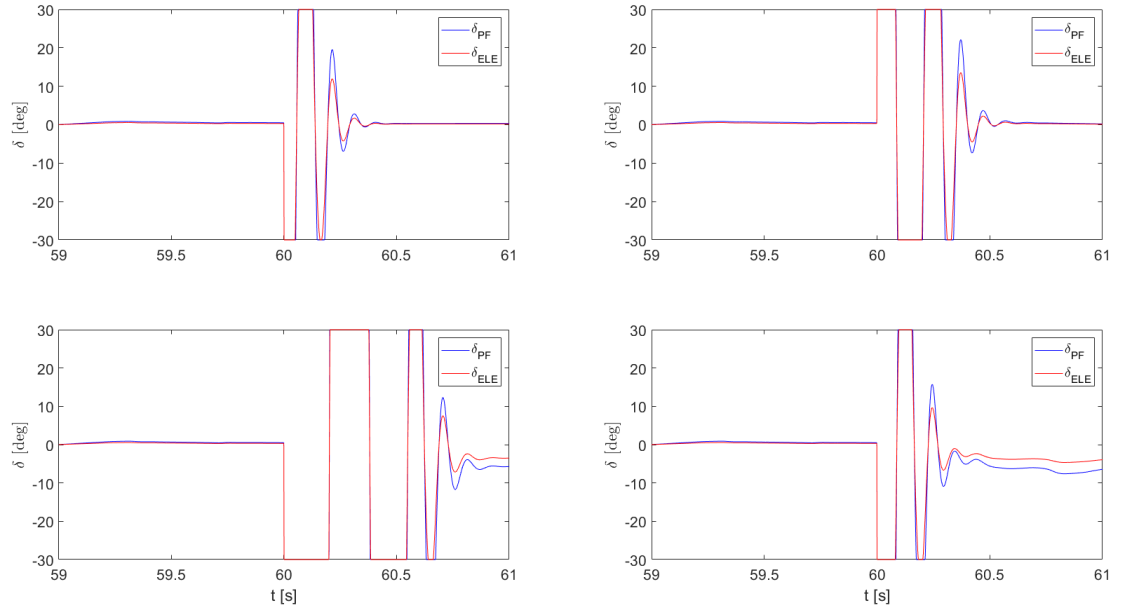


Figure (6.10) Zoomed in control policies for checking the robustness of the HDP angle of attack controller for different angle of attack reference signals for R1 case on top left, R2 case top right, R3 case bottom left and RC4 bottom right

6.5.2. Adaptability of the HDP controller

It was already shown that the HDP controller is quite robust to changes in the reference tracking signal. To investigate the adaptability of the control signal for a change in the reference angle of attack, the first and the two robust cases discussed in the previous section are chosen for checking the adaptability of the designed controller so that their performances can be compared to the robustness check case. In this case, online learning continues after the change in the reference signal. The behaviour of the controller and the state responses are plotted by Figure 6.11.

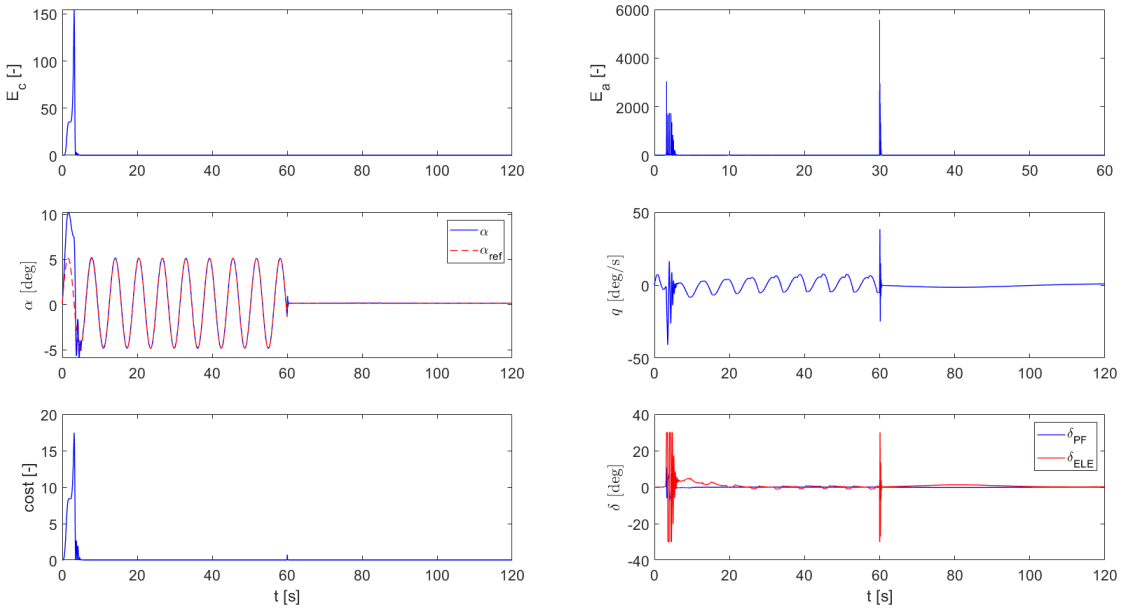


Figure (6.11) The results for checking the adaptability of the HDP controller for the constant angle of attack reference signals

Figure 6.11 shows that the angle of attack quickly adapts to the new constant signal. The zoomed graphs are shown by Figure 6.12 shows that precise tracking occurs less than half a second. Comparing the α response in this case with what was presented for robustness check indicates that there is not much a difference between the two cases. Therefore, in case of a constant signal, online learning does not necessarily change the angle of attack attitude.

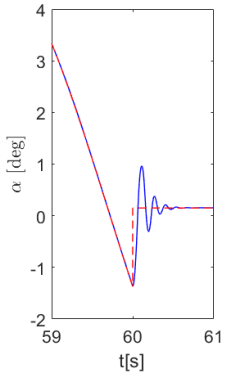


Figure (6.12) The zoomed results for checking the adaptability of the HDP controller for the constant angle of attack reference signals (blue (α), red(α_{ref}))

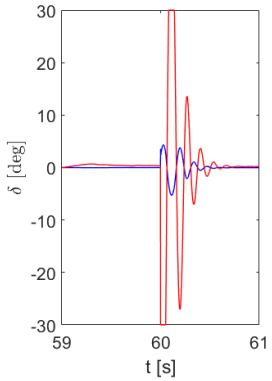


Figure (6.13) The effectors response in checking the adaptability for a second constant signal (blue (PF), red (ELE))

To see the effect of continuous online learning on the effectors' response for the case where a constant signal is given to the controller, Figure 6.13 is plotted. From the graph, it can be seen that the controller adapts to a new policy after half of a second and the policy shows fewer changes compared to the

robustness check case. It would be interesting to see the policy that the controller adopt in the two cases. Therefore, Figure 6.14 shows the policy of the controller for case R1 for robustness check and adaptability check for the last 10 seconds. It can be seen while the controller policies are different in the both cases, the robust controller achieved a better policy in terms of the contributing effect of the controllers, while the adaptive controller policy shows the antagonistic movements of the control surfaces while the controller utilization is less with the adaptive controller and appear as more stable.

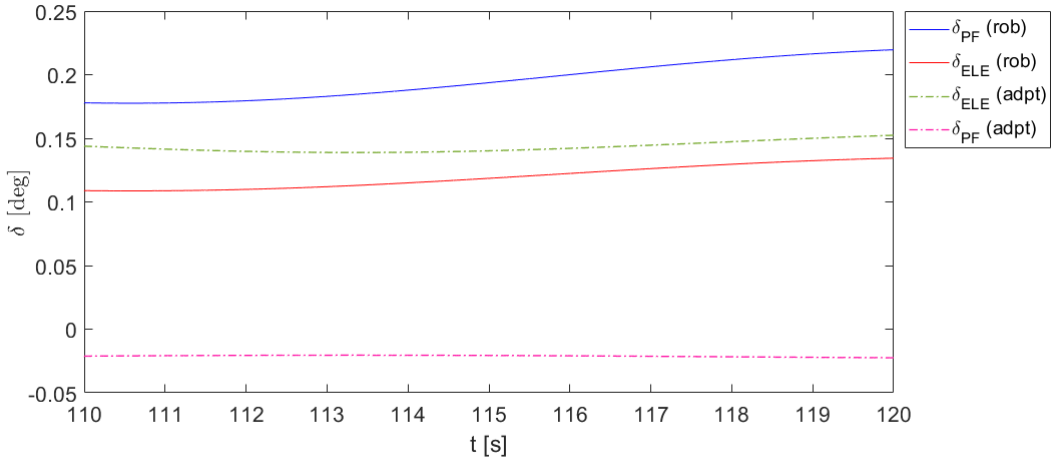


Figure (6.14) The control effectors deflection for robustness check and adaptability check for last 10 seconds of learning considering R1 case ('rob' denotes for robustness check case and 'adpt' corresponds to the adaptability check case)

The weights progress for the actor, critic and model networks are presented by Figure 6.15. The change in the reference signal results in a new update in the weights of the neural network and also small changes in the model output layer weights which corresponds to the fast adaptation.

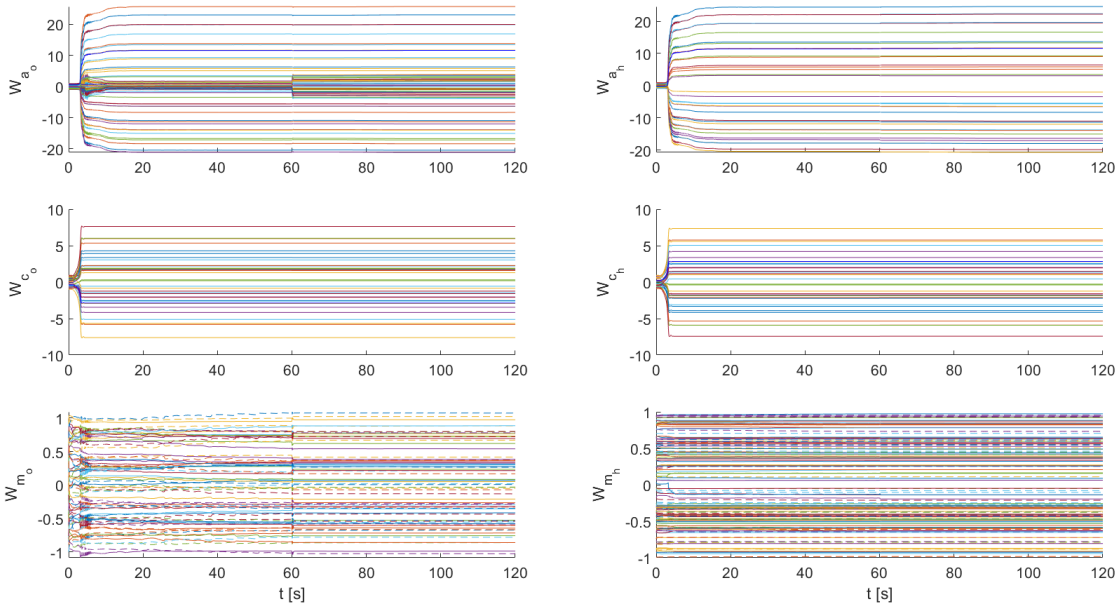


Figure (6.15) The network weight updates for R1 case in adaptation check

The next reference signal for checking the online adaptability of the controller is a sinusoidal signal with a different amplitude, but the same frequency. The controller again shows a high adaptation rate, as shown by plots in Figure 6.16. The actor error increases tremendously at the instant of the change while there is a peak in the cost values and a smaller one for the critic error. The error quickly becomes

low and close to zero with the actor network still show some minor errors during the online learning. There is also some deterioration of the pitch rate response in the first few seconds of signal change that will quickly change to a periodic reaction that would be expected for the angle of attack tracking and as can be seen the amplitude of the pitch rate response become bigger.

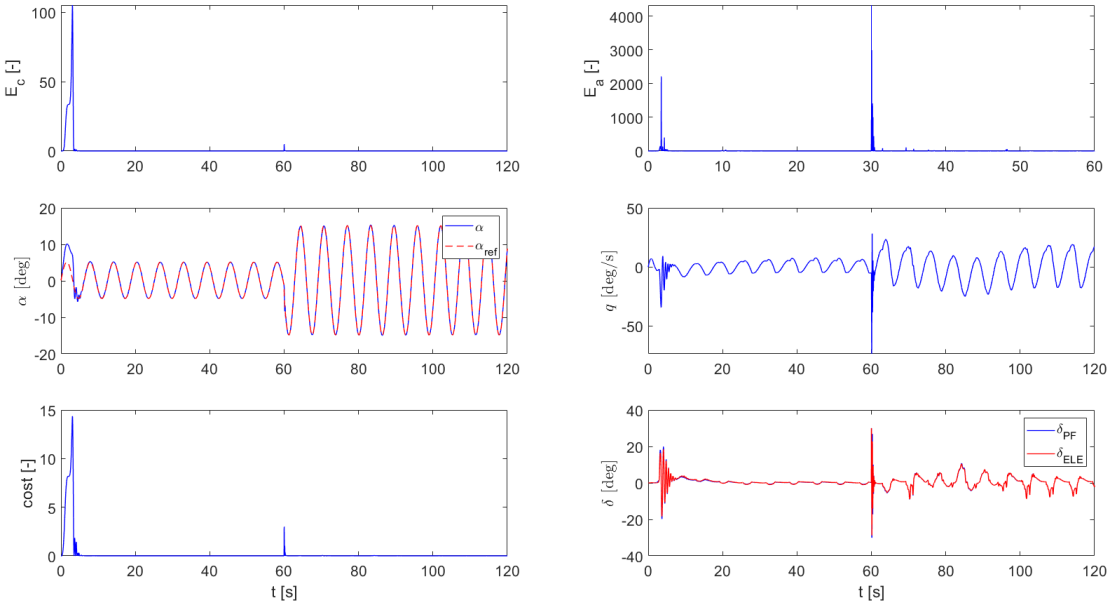


Figure (6.16) The results for checking the adaptability of the HDP controller for R2 case

The zoomed-in reference tracking behaviour is shown in the next plot in addition to the zoomed control policy adaptation. Looking at the graphs, it can be seen that the controller, in this case, can quickly adapt the new signal in less than half a second. Looking at the control policy adaptation, it can be seen that the controller new policy use both effectors in the same direction and therefore both pitch flaps and elevons are contributing to follow the angle of attack new reference signal, as the control deflections become more significant. There is a nice allocation of control between the two effectors.

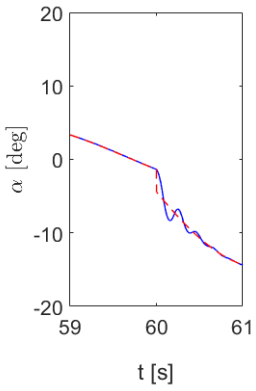


Figure (6.17) The zoomed results for checking the adaptability of the HDP controller for the R2 case (blue (α), red(α_{ref}))

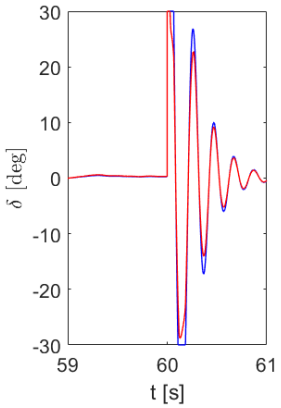


Figure (6.18) The zoomed effectors response in checking the adaptability of the HDP controller for R2 case (blue (PF), red (ELE))

To see the difference between the policy obtained in this case with the robustness check case, the effectors' response for the last 10 seconds are plotted for both cases by Figure 6.19. As one can see, the controller obtained a more consistent controller allocation for the adaptive controller, as the controller uses both controllers to the same extent. This is while for the robust controller, the pitch flaps are used more than elevons. However, in both cases, the effectors are used in a contributing way and therefore, adding to the control power for the angle of attack track.

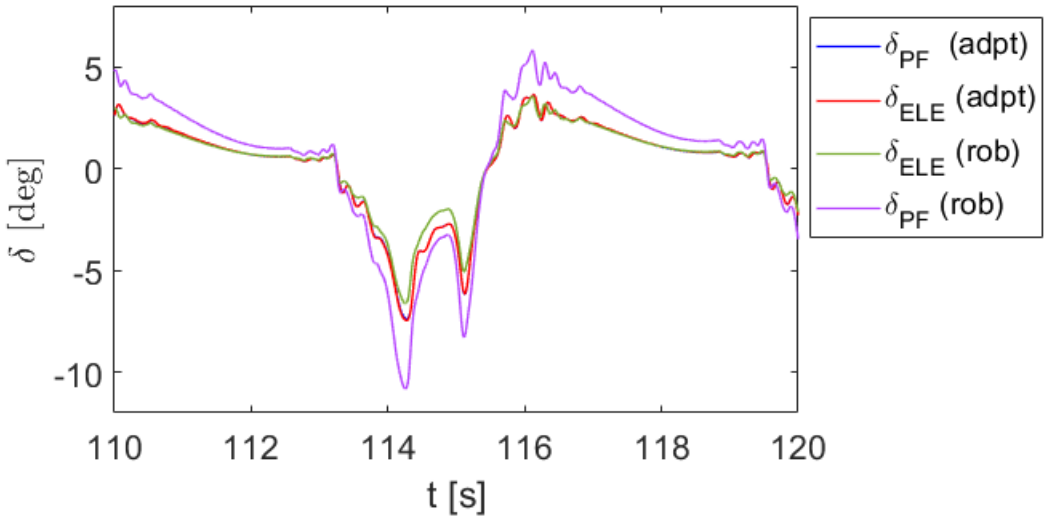


Figure (6.19) The control effectors deflection for robustness check and adaptability check for last 10 seconds of learning considering R2 case ('rob' denotes for robustness check case, and 'adpt' corresponds to the adaptability check case)

The weights updates for this case are shown by Figure 6.20. The weights progress for the actor shows constant improvements, and therefore, the policy is not converged within the second 60 seconds. So, the actor is still learning, while the critic weights remain unchanged even with the change in the reference signal. This comes from the low learning rate of the critic compared to the actor. The model weights are also changing, which is expected as the model is only identified locally.

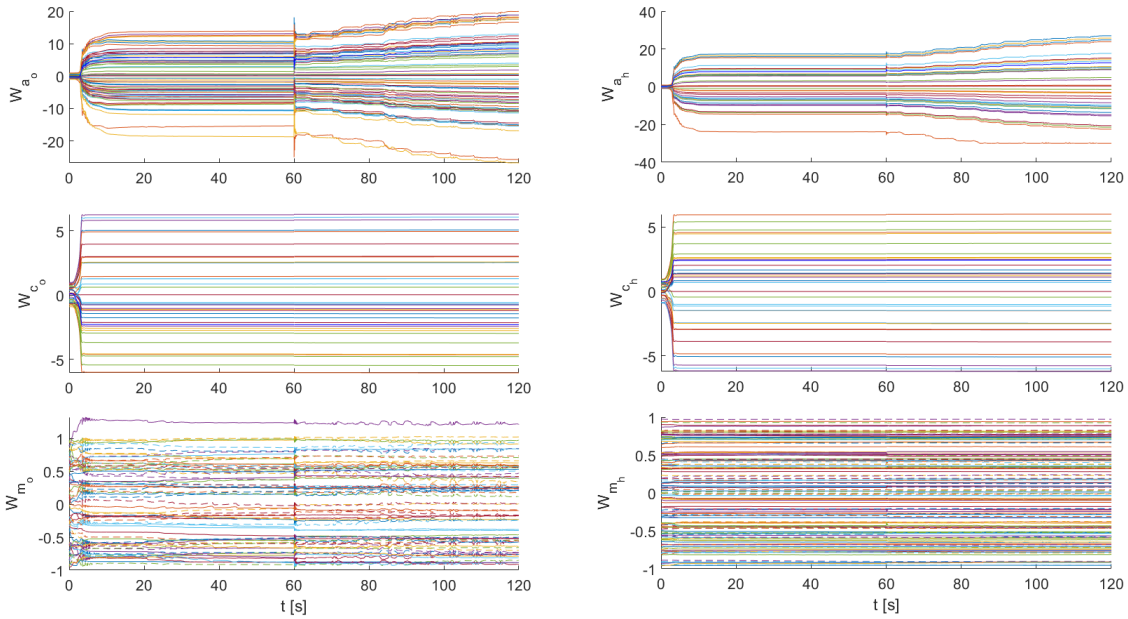


Figure (6.20) The weights progress for the case R2 in adaptability check

7

Conclusions and Next Steps

With the new advancements in automation, new challenges are defined for automatic control problems. These issues become even more emphasized when dealing with complex nonlinear systems such as a tailless fighter aircraft with innovative designs. One of the most promising approaches to deal with control challenges of unknown and with complex dynamics is adaptive control. This thesis report goal is set to the development of an online, optimal and adaptive continuous reinforcement learning-based controller for the nonlinear and non-conventional Innovative Control Effector (ICE) aircraft as also described by the formal research objective given by chapter 1. In this chapter, a conclusion is given based on the answers to the research questions that are mentioned in the following. These findings conclude this thesis master in the direction to achieve the research objective.

7.1. Literature Review Conclusion

The first part of this report includes the formulation of the research problem, the research objective and motivation and the research questions. In the second part, so the literature review and preliminary implementations, the goal is to answer the first two research questions of **RQ1: What is the state-of-the-art in the field of Adaptive and Robust control with Reinforcement Learning?** and **RQ2: Which control task is suitable for the ICE model to present the online adaptability and performance of the RL methods?**. Also, the research question of **RQ3: How can the performance of an approximate RL-based control system be compared to the Discrete RL-based controller?** is partially answered. The detail conclusion for each research question and proceeding sub-questions is given in the following. The next part of the report is written based on the answers to these questions.

RQ1 Sub-questions:

- What are the recent achievements in the subject of approximate reinforcement learning?
- What are the latest developments in approximate dynamic programming methods in the context of aerospace systems?
- What are the latest developments in actor-critic Design methods?
- What are the best function approximation methods for online and large states applications, in terms of data efficiency and computational costs?

An exploration of the state-of-the-art literature on RL problems and solution methods are given by chapter 2 to get insights on the general reinforcement learning problem and its location in the control field. The purpose is to identify the suitable continuous RL methods for getting closer to achieving the primary research objective. Therefore, the basis of RL algorithms and classifications is presented in this chapter. Within these methods, a distinction is made for continuous approaches that enable the controller to accept continuous state and actions. It is found that reinforcement learning problem,

in general, can be described in many terms and can be used for various applications. Therefore, the research needs to be narrowed down to approximate reinforcement learning application in the control field. The Approximate Dynamic Programming (ADP), which can be seen as the approximate reinforcement learning control problem, is still a growing field. One of the most challenging fields in ADP area of focus is online learning to achieve more accuracy in adaptation to, e.g. tracking tasks. One of the most promising frameworks within the ADP approaches, are ACD algorithms which have been developed from the most popular architecture of Heuristic Dynamic Programming (HDP) to more advanced structures of Global Dual Heuristic Dynamic Programming (GDHDP). Therefore, an overview of the Adaptive Critic Design (ACD) algorithms is given with their different architectures. Some of the structures that are formulated by ACD approach require a model learning method. It is found that in order to identify the model of the system in an online learning framework of ACD and also approximate other entities of the architecture, neural networks which are one of the most popular function approximators can be used. These FAs can adapt locally and therefore satisfy the data efficiency requirement and have fast learning capabilities.

Following up with the same research question, and in order to emphasize the location of reinforcement-learning based control in the field of aircraft Guidance, Navigation, and Control (GNC) and the context of adaptive and optimal control of air vehicles, the use of RL-based control in design of Flight Control Systems (FCSs) was discussed in chapter 3. In the same chapter, some of the most recent applications of approximate reinforcement learning-based control in aerospace vehicles are mentioned to show the promising features of such controllers. It was found that advanced FCSs are highly desired in the aviation community and especially in the automation field. Being an adaptive approach (dealing with unexpected faults) with the inherent ability to deal with constraints without any prior knowledge about the system, makes the approximate RL approaches such as ACDs a potential for improvements in current FCSs.

RQ2 Sub-questions:

- What are the objectives of the ICE design, in terms of its mission?
- What requirements these objectives would imply from the control perspective?
- What are the pros and cons of the methods used for the control problem of the over-actuated ICE model so far?

chapter 4 is aimed to answer the below subquestions on the ICE aircraft by giving a review on the ICE model and its components. It is shown that the non-conventional configuration of the ICE aircraft which equip the stealth features of the aircraft and the redundancy of the control effector which then compensate for high manoeuvrability, result in a challenging control problem including a complex control allocation problem. In the same chapter, one of the most recent applications of adaptive control approaches using RL is shown for the ICE aircraft tracking control problem, so the reinforcement learning approach of discrete Q-learning. It is found that discrete Q-learning method results in large oscillations around the reference trajectory in most of the cases and aggressive control policies in addition to its offline nature and high computational cost. It is expected that the approximate RL approach can solve the tracking control problem of ICE aircraft with less to no oscillations and in an online manner, without prior information about the global model of the aircraft and with less aggressive control policies to ensure the safety of the actuators. In addition, to answer the main research question 2 and for hence of comparison, the altitude tracking control task will be used to show the advantage of approximate RL approach over the already available discrete approach of Q-learning.

RQ3 Sub-questions:

- Is the tracking error less for the approximate RL-based controller?
- Is the designed controller faster in achieving the control objective?

In order to answer the above questions, the discrete Q-learning approach and one of the most popular and widely used ACD approach of heuristic dynamic programming is developed and successfully implemented on more straightforward but nonlinear control problems of the rotatory pendulum swing-up

and stabilization and missile tracking control problems, respectively. The case study of a nominal and under-actuated swing-up pendulum and a more complex over-actuated swing-up pendulum problem are used to show the ability of RL in dealing with control allocation, solving a regulation problem and also following a reference signal. The HDP implementation of nonlinear missile tracking control problem confirms that HDP is suitable for learning and performing nonlinear control tasks. For both problems, the hyper-parameters are tuned using trial and error method, till the convergence of the controller to a (near) optimal policy. In addition, it is found that by adding noise to the control input at the beginning of the learning, the online learning improves and the (near) optimal policy is achieved much faster and in the very first few seconds. Although the two problems are totally different systems, however, it is observed that the HDP controller results in control policies that are smoother. The HDP method proposed here is as a basis for solving the complex tracking control problem of ICE aircraft in the next part of the report.

7.2. Main Conclusion

Approximate RL approach of HDP is developed to solve the ICE aircraft attitude tracking control problem for the objective of attitude and altitude control. In the central part of this thesis, two last research questions and the remaining part of the third research questions are answered. These questions are **RQ4: How is the performance of the controller designed by continuous RL methods influenced by the hyper-parameters?** and **RQ5: How adaptive and robust is the approximate RL-based FCS to changes?**. In the following, these questions are discussed separately by also presenting the sub-questions.

To answer what is left from RQ3, the HDP continuous controller is implemented for attitude tracking initially. Results show that the HDP controller can learn online to control ICE aircraft longitudinal dynamics as to follow a reference angle of attack trajectory with no prior knowledge about the dynamics of the aircraft. The controller also shows that altitude stabilization around the desired trajectory is possible with using a PID outer loop. The results are evaluated for one or higher number of symmetrical sets of control surfaces by just providing tracking error as the information provided to the controller. It is shown that the controller can follow the reference signal precisely and quickly. For the altitude tracking problem, while the objective is met, however, the policy that the controller gains use control effectors antagonistically and therefore result in excess drag. By just looking at the results, it is found that the performance of the HDP controller is improved significantly than that of the discrete RL controller for the ICE aircraft, both in terms of tracking performance and the smoothness of the control policy in addition to the online learning possibility. The inefficient utilization of the control effectors, however, can be improved by the inclusion of the effector utilizations in the cost function.

RQ4 Sub-questions:

- How function approximation parameters affect RL controller?
- How learning parameters affect RL controller?

To answer this research question, a sensitivity analysis on some of the hyper-parameters of the controller, including the number of neurons, the learning rates and networks weights initialization is performed to see their effect on the stability and convergence of the controller, the tracking error and the performance of the RL algorithm as well as the control effort for the effectors. It is shown that these parameters affect the RL controller performance and stability. However, the most significant effect is found about the weight initialization of the networks as the success ratio of the controller increased by 100 % by just increasing the range of the random weight initializations. Other parameters effect are found to be less (but not necessarily minor) for the case of the ICE aircraft longitudinal control.

RQ5 Sub-questions:

- How robust is the online learned controller to change in the reference signal?
- How adaptive is the controller to changes in the reference signal?

The robustness and adaptability of the controller are tested by changing the reference signal for the online learned controller after some time without and with online learning still performing. The behaviour of the ICE aircraft and the control policy changes are observed for constant and different in amplitude and frequency signals as well as a different in the nature reference signal. It is found that in all the cases, the algorithm is robust and adaptive by performing a rapid adaptation to the new reference signal (less than a second). It is shown that if online learning continues with the introduction of the new signal, the controller achieves a better policy.

Until now, the control problem of the ICE aircraft tailless configuration is considered by literature very challenging. Some of the control approaches, such as Incremental Nonlinear Control Allocation (INCA), have proven to be very successful. Although these methods are mostly model-based, their integration with the model-free approaches of RL can result in a more powerful controller. To reduce the significant initial errors induced by neural networks approximations, incremental methods such as Incremental Heuristic Dynamic Programming (IHDP) or other Adaptive Critic Design (ACD) algorithms that have shown promising performances, such as Dual Heuristic Dynamic Programming (DHP) can be implemented for the ICE aircraft. The effect of adding biases to the neural networks structure can be investigated if the aircraft start at trim positions. Also, the adaptability of the controller to faults in the system or change in the environment can be discussed for the proposed approach (for example if one of the effectors become unavailable). The comparison of the continuous RL-based controller with the model-based approaches is a challenging field of study. Performance measures in model-based are based on the error between the deflections of the effectors and a desired valued obtained from the model. However, for an RL controller, no instruction command is available, since it's a goal-oriented approach. Regarding the adaptability of the controller, online learning is a challenging task to be obtained purely with RL. Some of these limitations can be tackled with a combination of RL with model-based methods such as Incremental Heuristic Dynamic Programming (IHDP) and Incremental Dual Heuristic programming (IDHP) methods. This report focused on the longitudinal control of the ICE aircraft. However, the lateral-directional control problem RL solution is still unknown to researchers. Therefore, this aspect of the ICE control problem should also be investigated using RL approaches. The optimal control of the ICE aircraft in this study focuses on the optimal tracking control and not optimal control allocation. As the large control suite of the ICE requires an optimal control allocation solution in addition to the optimal reference tracking task, this field should also be further investigated with RL control.

Bibliography

[1] K. Doya, *Reinforcement learning in continuous time and space,* Neural computation **12**, 219 (2000).

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction "Complete Draft"* (MIT press Cambridge, 2018).

[3] J. Ding and S. Balakrishnan, *Approximate dynamic programming solutions with a single network adaptive critic for a class of nonlinear systems,* Journal of Control Theory and Applications **9**, 370 (2011).

[4] E.-J. van Kampen, Q. Chu, and J. Mulder, *Continuous adaptive critic flight control aided with approximated plant dynamics,* in *AIAA Guidance, Navigation, and Control Conference and Exhibit* (2006) p. 6429.

[5] S. Ferrari, J. E. Steck, and R. Chandramohan, *Adaptive feedback control by constrained approximate dynamic programming,* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **38**, 982 (2008).

[6] P. S. de Vries, *Reinforcement Learning-based Control Allocation for the ICE Aircraft Model*, Tech. Rep. (Delft University of Technology, 2017).

[7] K. M. Dorsett and D. R. Mehl, *Innovative control effectors (ICE)*, Tech. Rep. (Lockheed Martin Tactical Aircraft Systems Fort Worth Tx, 1996).

[8] M. A. Niestroy, K. M. Dorsett, and K. Markstein, *A tailless fighter aircraft model for control-related research and development,* in *AIAA Modeling and Simulation Technologies Conference* (2017) p. 1757.

[9] Y. Zhou, *Online reinforcement learning control for aerospace systems*, Ph.D. thesis, Delft University of Technology (2018).

[10] J. Bowlus, D. Multhopp, S. Banda, J. Bowlus, D. Multhopp, and S. Banda, *Challenges and opportunities in tailless aircraft stability and control,* in *Guidance, Navigation, and Control Conference* (1997) p. 3830.

[11] R. Whitford, *Design for air combat,* RUSI Journal **132**, 79 (1987).

[12] S. Thrun and M. L. Littman, *Reinforcement learning: an introduction,* AI Magazine **21**, 103 (2000).

[13] 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).

[14] Y. Zhou, E.-J. van Kampen, and Q. P. Chu, *Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming,* (2017).

[15] Y. Zhou, E.-J. van Kampen, and Q. P. Chu, *Incremental model based online dual heuristic programming for nonlinear adaptive control,* Control Engineering Practice **73**, 13 (2018).

[16] S. Ferrari and R. F. Stengel, *Online adaptive critic flight control,* Journal of Guidance, Control, and Dynamics **27**, 777 (2004).

[17] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control* (John Wiley & Sons, 2012).

[18] W. B. Powell, *What you should know about approximate dynamic programming,* Naval Research Logistics (NRL) **56**, 239 (2009).

[19] S. J. Bradtke and A. G. Barto, *Linear least-squares algorithms for temporal difference learning,* Machine learning **22**, 33 (1996).

[20] W. Eerland, C. de Visser, and E.-J. van Kampen, *On approximate dynamic programming with multivariate splines for adaptive control,* arXiv preprint arXiv:1606.09383 (2016).

[21] T. Lombaerts, *Automatic Flight Control Systems–Latest Developments* (InTech, 2011).

[22] I. Matamoros Cid, *Nonlinear control allocation for a high-performance tailless aircraft with innovative control effectors: An incremental robust approach,* (2017).

[23] R. S. Sutton and A. G. Barto, *Toward a modern theory of adaptive networks: expectation and prediction.* Psychological review **88**, 135 (1981).

[24] E. Van Kampen, Q. Chu, and J. Mulder, *Online adaptive critic flight control using approximated plant dynamics,* in *2006 International Conference on Machine Learning and Cybernetics*, Vol. 1 (2006).

[25] S. Schaal, *Learning from demonstration,* in *Advances in neural information processing systems* (1997) pp. 1040–1046.

[26] B. Farely and W. Clark, *Simulation of self-organizing systems by digital computers,* IEEE Transactions of Professional Group of Information Theory, PGIT-4 , 76 (1954).

[27] M. Minsky, *Steps toward artificial intelligence,* Proceedings of the IRE **49**, 8 (1961).

[28] D. Michie, *Experiments on the mechanization of game-learning part i. characterization of the model and its parameters,* The Computer Journal **6**, 232 (1963).

[29] D. Michie and R. A. Chambers, *Boxes: An experiment in adaptive control,* Machine intelligence **2**, 137 (1968).

[30] A. G. Barto, R. S. Sutton, and C. W. Anderson, *Neuronlike adaptive elements that can solve difficult learning control problems,* IEEE transactions on systems, man, and cybernetics , 834 (1983).

[31] R. S. Sutton, *Temporal credit assignment in reinforcement learning,* (1984).

[32] B. Widrow and F. W. Smith, *Pattern-recognizing control systems,* Computer and Information Sciences (COINS) Proceedings (1964).

[33] B. Widrow, N. K. Gupta, and S. Maitra, *Punish/reward: Learning with a critic in adaptive threshold systems,* IEEE Transactions on Systems, Man, and Cybernetics , 455 (1973).

[34] M. Tsetlin, *Automaton theory and modeling of biological systems, volume 102 of Mathemat ics in Science and Engineering* (Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, 1973).

[35] J. Holland, *Adaptation in natural and artificial systems: an introductory analysis with application to biology,* Control and artificial intelligence (1975).

[36] A. H. Klopf, *Brain function and adaptive systems: a heterostatic theory*, Tech. Rep. (AIR FORCE CAMBRIDGE RESEARCH LABS HANSCOM AFB MA, 1972).

[37] C. W. Anderson, *Learning to control an inverted pendulum using neural networks,* IEEE Control Systems Magazine **9**, 31 (1989).

[38] C. J. Watkins and P. Dayan, *Q-learning,* Machine learning **8**, 279 (1992).

[39] P. J. Werbos, *Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research,* IEEE Transactions on Systems, Man, and Cybernetics **17**, 7 (1987).

[40] N. Ravishankar and M. Vijayakumar, *Reinforcement learning algorithms: survey and classification,* Indian Journal of Science and Technology **10** (2017).

[41] I. Grondman, *Online model learning algorithms for actor-critic control,* (2015).

[42] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, Vol. 1 (MIT press Cambridge, 1998).

[43] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, *Approximate reinforcement learning: An overview,* in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on* (IEEE, 2011) pp. 1–8.

[44] R. S. Sutton, *Learning to predict by the methods of temporal differences,* Machine learning **3**, 9 (1988).

[45] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach* (Malaysia; Pearson Education Limited,, 2016).

[46] P. Y. Glorennec, *Reinforcement learning: An overview,* in *Proceedings European Symposium on Intelligent Techniques (ESIT-00), Aachen, Germany* (Citeseer, 2000) pp. 14–15.

[47] L.-J. Lin, *Self-improving reactive agents based on reinforcement learning, planning and teaching,* Machine learning **8**, 293 (1992).

[48] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, *Optimal and autonomous control using reinforcement learning: A survey,* IEEE transactions on neural networks and learning systems **29**, 2042 (2018).

[49] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, *An analysis of reinforcement learning with function approximation,* in *Proceedings of the 25th international conference on Machine learning* (ACM, 2008) pp. 664–671.

[50] R. Schoknecht, *Optimality of reinforcement learning algorithms with linear function approximation,* in *Advances in neural information processing systems* (2003) pp. 1587–1594.

[51] V. R. Konda and J. N. Tsitsiklis, *Onactor-critic algorithms,* SIAM journal on Control and Optimization **42**, 1143 (2003).

[52] H. R. Berenji and D. Vengerov, *A convergent actor-critic-based frl algorithm with application to power management of wireless transmitters,* IEEE Transactions on Fuzzy Systems **11**, 478 (2003).

[53] L. Buşoniu, A. Daniels, and R. Babuška, *Online learning for optimistic planning,* Engineering Applications of Artificial Intelligence **55**, 70 (2016).

[54] M. Riedmiller, J. Peters, and S. Schaal, *Evaluation of policy gradient methods and variants on the cart-pole benchmark,* in *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on* (IEEE, 2007) pp. 254–261.

[55] C. J. C. H. Watkins, *Learning from delayed rewards*, Ph.D. thesis, King's College, Cambridge (1989).

[56] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, *Adaptive linear quadratic control using policy iteration,* in *Proceedings of the American control conference*, Vol. 3 (Citeseer, 1994) pp. 3475–3475.

[57] R. Islam and R. Seraj, *Unifying on-policy and off-policy learning in td learning and actor-critic methods,* (2017).

[58] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*, Vol. 37 (University of Cambridge, Department of Engineering Cambridge, England, 1994).

[59] V. R. Konda and J. N. Tsitsiklis, *Actor-critic algorithms,* in *Advances in neural information processing systems* (2000) pp. 1008–1014.

[60] L. Baird, *Residual algorithms: Reinforcement learning with function approximation,* in *Machine Learning Proceedings 1995* (Elsevier, 1995) pp. 30–37.

[61] G. J. Gordon, *Stable function approximation in dynamic programming,* in *Machine Learning Proceedings 1995* (Elsevier, 1995) pp. 261–268.

[62] J. N. Tsitsiklis and B. Van Roy, *Feature-based methods for large scale dynamic programming,* Machine Learning **22**, 59 (1996).

[63] R. J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning,* Machine learning **8**, 229 (1992).

[64] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, *Policy gradient methods for reinforcement learning with function approximation,* in *Advances in neural information processing systems* (2000) pp. 1057–1063.

[65] J. Peters, K. Mülling, and Y. Altun, *Relative entropy policy search.* in *AAAI* (Atlanta, 2010) pp. 1607–1612.

[66] S. Adam, L. Busoniu, and R. Babuska, *Experience replay for real-time reinforcement learning control,* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**, 201 (2012).

[67] A. W. Moore and C. G. Atkeson, *Prioritized sweeping: Reinforcement learning with less data and less time,* Machine learning **13**, 103 (1993).

[68] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, *Sample efficient actor-critic with experience replay,* arXiv preprint arXiv:1611.01224 (2016).

[69] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, *Asynchronous methods for deep reinforcement learning,* in *International Conference on Machine Learning* (2016) pp. 1928–1937.

[70] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, *Trust region policy optimization,* in *International Conference on Machine Learning* (2015) pp. 1889–1897.

[71] R. S. Sutton, *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,* in *Machine Learning Proceedings 1990* (Elsevier, 1990) pp. 216–224.

[72] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, *Pac model-free reinforcement learning,* in *Proceedings of the 23rd international conference on Machine learning* (ACM, 2006) pp. 881–888.

[73] M. Dorigo and M. Colombetti, *Robot shaping: Developing autonomous agents through learning,* Artificial intelligence **71**, 321 (1994).

[74] J. Randløv and P. Alstrøm, *Learning to drive a bicycle using reinforcement learning and shaping.* in *ICML*, Vol. 98 (Citeseer, 1998) pp. 463–471.

[75] A. Y. Ng, D. Harada, and S. Russell, *Policy invariance under reward transformations: Theory and application to reward shaping,* in *ICML*, Vol. 99 (1999) pp. 278–287.

[76] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*, Vol. 1 (Athena scientific Belmont, MA, 1995).

[77] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro, *A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots,* in *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (IEEE, 2013) pp. 4896–4901.

[78] R. Glaubius, M. Namihira, and W. D. Smart, *Speeding up reinforcement learning using manifold representations: Preliminary results,* in *Proceedings of the IJCAI 2005 Workshop on Reasoning with Uncertainty in Robotics (RUR 05)* (2005).

[79] C. Gaskett, D. Wettergreen, and A. Zelinsky, *Q-learning in continuous state and action spaces,* in *Australasian Joint Conference on Artificial Intelligence* (Springer, 1999) pp. 417–428.

[80] A. Naruta, *Continuous state and action q-learning framework applied to quadrotor uav control,* Master Thesis (2017).

[81] R. Bellman, *Dynamic programming* (Courier Corporation, 2013).

[82] T. J. Perkins and A. G. Barto, *Lyapunov design for safe reinforcement learning,* Journal of Machine Learning Research **3**, 803 (2002).

[83] J. K. Peterson, *On-line estimation of the optimal value function: Hjb-estimators,* in *Advances in neural information processing systems* (1993) pp. 319–326.

[84] R. Munos, *A convergent reinforcement learning algorithm in the continuous case based on a finite difference method,* in *IJCAI (2)* (1997) pp. 826–831.

[85] R. Munos and P. Bourgine, *Reinforcement learning for continuous stochastic control problems,* in *Advances in neural information processing systems* (1998) pp. 1029–1035.

[86] P. Dayan and S. P. Singh, *Improving policies without measuring merits,* in *Advances in neural information processing systems* (1996) pp. 1059–1065.

[87] D. Liu, X. Yang, D. Wang, and Q. Wei, *Reinforcement-learning-based robust controller design for continuous-time uncertain nonlinear systems subject to input constraints,* IEEE transactions on cybernetics **45**, 1372 (2015).

[88] 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).

[89] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*, Vol. 39 (CRC press, 2010).

[90] P. Werbos, *Approximate dynamic programming for realtime control and neural modelling,* Handbook of intelligent control: neural, fuzzy and adaptive approaches , 493 (1992).

[91] D. V. Prokhorov and D. C. Wunsch, *Adaptive critic designs,* IEEE transactions on Neural Networks **8**, 997 (1997).

[92] H. Modares, M.-B. N. Sistani, and F. L. Lewis, *A policy iteration approach to online optimal control of continuous-time constrained-input systems,* ISA transactions **52**, 611 (2013).

[93] T. Xie, H. Yu, and B. Wilamowski, *Comparison of fuzzy and neural systems for implementation of nonlinear control surfaces,* in *Human–Computer Systems Interaction: Backgrounds and Applications 2* (Springer, 2012) pp. 313–324.

[94] K. G. Vamvoudakis and F. L. Lewis, *Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem,* Automatica **46**, 878 (2010).

[95] E. H. Mamdani, *Application of fuzzy algorithms for control of simple dynamic plant,* in *Proceedings of the institution of electrical engineers*, Vol. 121 (IET, 1974) pp. 1585–1588.

[96] M. Sugeno, *An introductory survey of fuzzy control,* Information sciences **36**, 59 (1985).

[97] 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).

[98] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning,* arXiv preprint arXiv:1509.02971 (2015).

[99] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, *Deterministic policy gradient algorithms,* in *ICML* (2014).

[100] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, *Handbook of learning and approximate dynamic programming*, Vol. 2 (John Wiley & Sons, 2004).

[101] R. Enns and J. Si, *Helicopter trimming and tracking control using direct neural dynamic programming,* IEEE Transactions on Neural Networks **14**, 929 (2003).

[102] A. Helmer, C. C. de Visser, and E.-J. Van Kampen, *Flexible heuristic dynamic programming for reinforcement learning in quad-rotors,* in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace* (2018) p. 2134.

[103] R. F. Stengel and S. Ferrari, *3 model-based adaptive critic designs,* Handbook of learning and approximate dynamic programming **2**, 65 (2004).

[104] D. V. Prokhorov, R. A. Santiago, and D. C. Wunsch II, *Adaptive critic designs: A case study for neurocontrol,* Neural Networks **8**, 1367 (1995).

[105] 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).

[106] B. Kiumarsi and F. L. Lewis, *Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems,* IEEE transactions on neural networks and learning systems **26**, 140 (2015).

[107] B. Kiumarsi, W. Kang, and F. L. Lewis, *H∞ control of nonaffine aerial systems using off-policy reinforcement learning,* Unmanned Systems **4**, 51 (2016).

[108] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, *Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems,* Automatica **50**, 193 (2014).

[109] S. Balakrishnan, J. Ding, and F. L. Lewis, *Issues on stability of adp feedback controllers for dynamical systems,* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **38**, 913 (2008).

[110] X. Yang, D. Liu, D. Wang, and H. Ma, *Constrained online optimal control for continuous-time nonlinear systems using neuro-dynamic programming,* in *Control Conference (CCC), 2014 33rd Chinese* (IEEE, 2014) pp. 8717–8722.

[111] S. A. Fjerdingen and E. Kyrkjebø, *Safe reinforcement learning for continuous spaces through lyapunov-constrained behavior,* (2011).

[112] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, *Safe model-based reinforcement learning with stability guarantees,* in *Advances in Neural Information Processing Systems* (2017) pp. 908–918.

[113] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, *A lyapunov-based approach to safe reinforcement learning,* arXiv preprint arXiv:1805.07708 (2018).

[114] A. Kumar and R. Sharma, *Fuzzy lyapunov reinforcement learning for non linear systems,* ISA transactions **67**, 151 (2017).

[115] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction,* (2011).

[116] L. Busoniu, D. Ernst, R. Babusku, and B. De Schutter, *Exploiting policy knowledge in online least-squares policy iteration: An empirical study,* Automation, Computers, Applied Mathematics **19**, 521 (2010).

[117] A. Gosavi, *Simulation-based optimization,* parametric optimization techniques and reinforcement learning (2003).

[118] R. S. Sutton, A. G. Barto, and R. J. Williams, *Reinforcement learning is direct adaptive optimal control,* IEEE Control Systems **12**, 19 (1992).

[119] S. Bhasin, *Reinforcement learning and optimal control methods for uncertain nonlinear systems* (University of Florida, 2011).

[120] D. Vrabie and F. Lewis, *Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems,* Neural Networks **22**, 237 (2009).

[121] J. K. Williams, *Reinforcement learning of optimal controls,* in *Artificial intelligence methods in the environmental sciences* (Springer, 2009) pp. 297–327.

[122] D. P. Bertsekas, *Neuro-dynamic programming: An overview and recent results,* in *Operations Research Proceedings 2006* (Springer, 2007) pp. 71–72.

[123] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, *Model-free q-learning designs for linear discrete-time zero-sum games with application to h-infinity control,* Automatica **43**, 473 (2007).

[124] 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).

[125] P. E. Stingu and F. L. Lewis, *Adaptive dynamic programming applied to a 6dof quadrotor,* in *Computational Modeling and Simulation of Intellect: Current State and Future Perspectives* (IGI Global, 2011) pp. 102–130.

[126] J. Kober, J. A. Bagnell, and J. Peters, *Reinforcement learning in robotics: A survey,* The International Journal of Robotics Research **32**, 1238 (2013).

[127] J. Betts and I. Kolmanovsky, *Practical methods for optimal control using nonlinear programming,* Applied Mechanics Reviews **55**, B68 (2002).

[128] R. Freeman and P. Kokotovic, *Optimal nonlinear controllers for feedback linearizable systems,* in *American Control Conference, Proceedings of the 1995*, Vol. 4 (IEEE, 1995) pp. 2722–2726.

[129] J. Y. Lee, J. B. Park, and Y. H. Choi, *Integral q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems,* Automatica **48**, 2850 (2012).

[130] P. He and S. Jagannathan, *Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints,* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **37**, 425 (2007).

[131] T. Dierks and S. Jagannathan, *Online optimal control of nonlinear discrete-time systems using approximate dynamic programming,* Journal of Control Theory and Applications **9**, 361 (2011).

[132] Y. Luo and H. Zhang, *Approximate optimal control for a class of nonlinear discrete-time systems with saturating actuators,* Progress in Natural Science **18**, 1023 (2008).

[133] D. Vrabie, F. Lewis, and D. Levine, *Neural network-based adaptive optimal controller–a continuous-time formulation,* in *International Conference on Intelligent Computing* (Springer, 2008) pp. 276–285.

[134] X. Luo and J. Si, *Stability of direct heuristic dynamic programming for nonlinear tracking control using pid neural network,* in *Neural Networks (IJCNN), The 2013 International Joint Conference on* (IEEE, 2013) pp. 1–7.

[135] R. Chandramohan, J. Steck, K. Rokhsaz, and S. Ferrari, *Adaptive critic flight control for a general aviation aircraft: Simulations for the beech bonanza fly-by-wire test bed,* in *AIAA Infotech@ Aerospace 2007 Conference and Exhibit* (2007) p. 2795.

[136] M. Fliess and C. Join, *Model-free control,* International Journal of Control **86**, 2228 (2013).

[137] T. L. Vincent and W. J. Grantham, *Nonlinear and optimal control systems* (John Wiley & Sons, 1997).

[138] S. Balakrishnan and V. Biega, *Adaptive-critic-based neural networks for aircraft optimal control,* Journal of Guidance, Control, and Dynamics **19**, 893 (1996).

[139] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, *An application of reinforcement learning to aerobatic helicopter flight,* in *Advances in neural information processing systems* (2007) pp. 1–8.

[140] H. J. Kim, M. I. Jordan, S. Sastry, and A. Y. Ng, *Autonomous helicopter flight via reinforcement learning,* in *Advances in neural information processing systems* (2004) pp. 799–806.

[141] P. Abbeel and A. Y. Ng, *Apprenticeship learning via inverse reinforcement learning,* in *Proceedings of the twenty-first international conference on Machine learning* (ACM, 2004) p. 1.

[142] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, *Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof,* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **38**, 943 (2008).

[143] J. Buffington and J. Buffington, *Tailless aircraft control allocation,* in *Guidance, Navigation, and Control Conference* (1997) p. 3605.

[144] W. J. Gillard, *Innovative control effectors (configuration 101) dynamic wind tunnel test report. Rotary balance and forced oscillation tests*, Tech. Rep. (AIR FORCE RESEARCH LAB WRIGHT-PATTERSON AFB OH AIR VEHICLES DIRECTORATE, 1998).

[145] R. Eberhardt and D. Ward, *Indirect adaptive flight control of a tailless fighter aircraft,* in *Guidance, Navigation, and Control Conference and Exhibit* (1999) p. 4042.

[146] A. Ngo, W. Reigelsperger, S. Banda, and J. Bessolo, *Multivariable control law design for a tailless airplane,* in *Guidance, Navigation, and Control Conference* (1996) p. 3866.

[147] J. F. Buffington, *Modular control law design for the innovative control effectors (ICE) tailless fighter aircraft configuration 101-3*, Tech. Rep. (AIR FORCE RESEARCH LAB WRIGHT-PATTERSON AFB OH AIR VEHICLES DIRECTORATE, 1999).

[148] G. A. Addington and J. H. Myatt, *Control-surface deflection effects on the innovative control effectors (ICE 101) design*, Tech. Rep. (AIR FORCE RESEARCH LAB WRIGHT-PATTERSON AFB OH AIR VEHICLES DIRECTORATE, 2000).

[149] I. V. van der Peijl, *Physical splines for aerodynamic modelling of innovative control effectors,* (2017).

[150] W. Gillard, K. Dorsett, W. Gillard, and K. Dorsett, *Directional control for tailless aircraft using all moving wing tips,* in *22nd Atmospheric Flight Mechanics Conference* (2006) p. 3487.

[151] D. Kennedy and J. Conlon, *Inverted pendulum swing up controller,* (2011).

[152] 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).

[153] D. Kroezen, *Online reinforcement learning for flight control: An adaptive critic design without prior model knowledge,* (2019).

Appendices

A

Additional Results for ICE Aircraft

Additional results for the attitude control benchmarks are shown here. They quantitatively discussed by the tables in the conference paper and the plots are presented here.

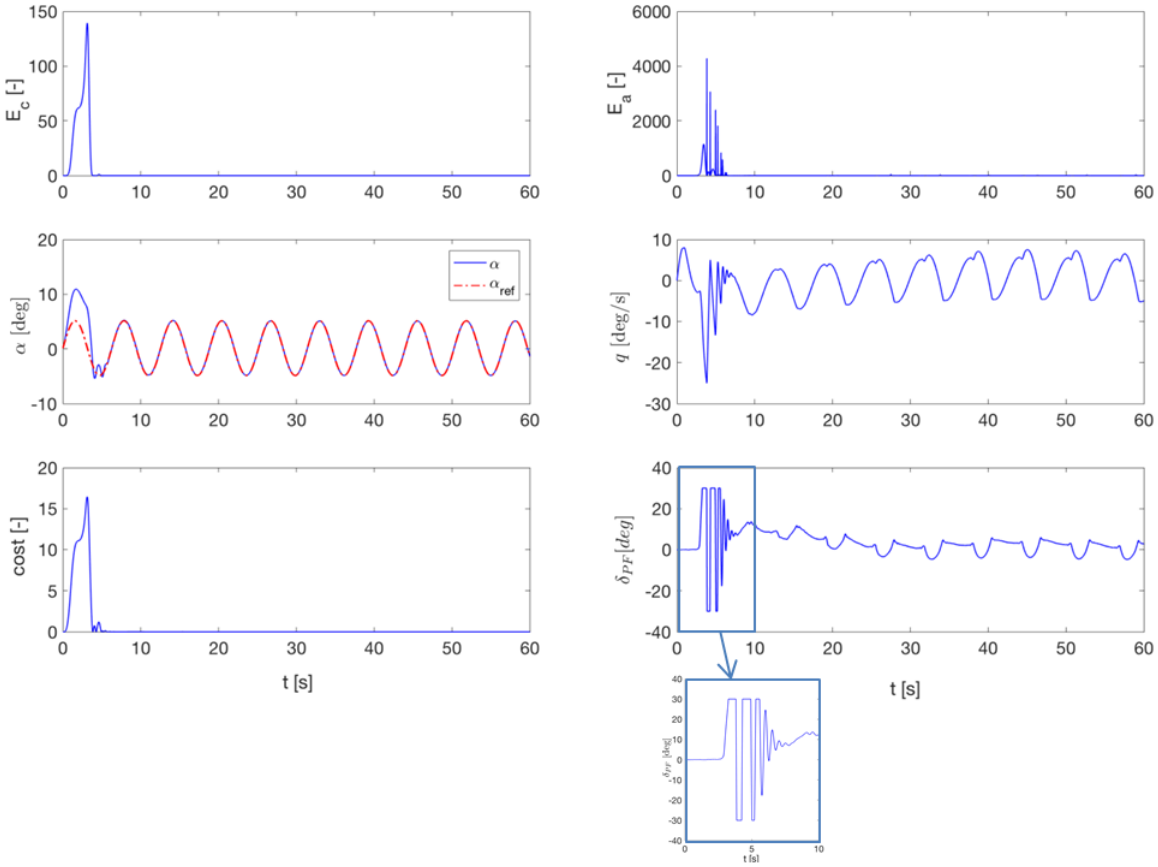


Figure (A.1) The attitude control results for BC1 case with critic network error in top left, the actor network error in top right, the angle of attack response and the reference angle of attack in middle left, the pitch rate response in middle right, the cost value in bottom left, the pf deflection in bottom right

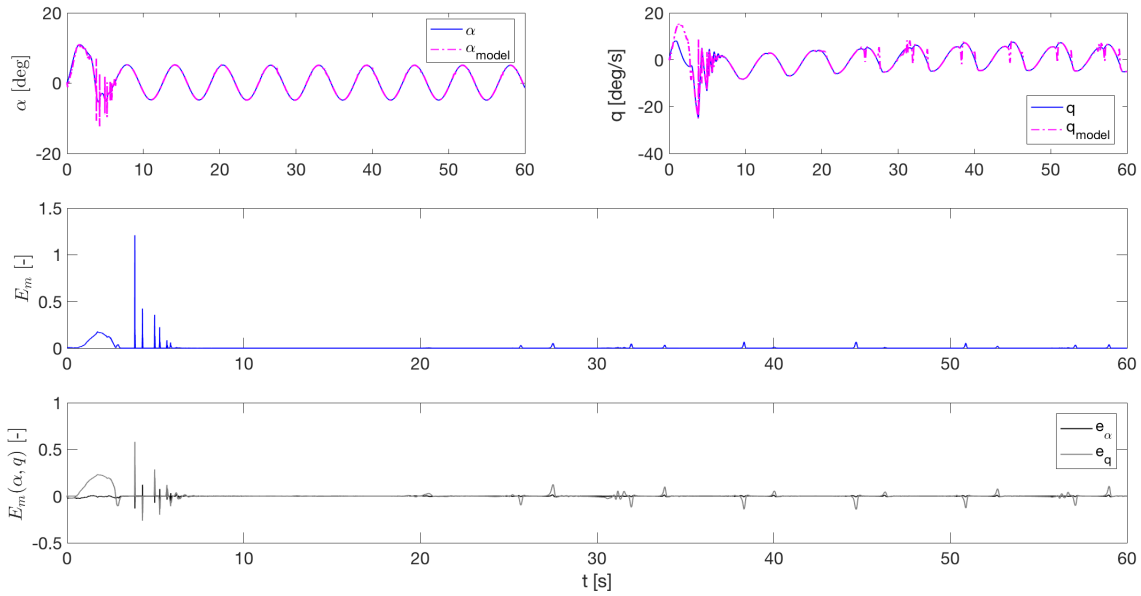


Figure (A.2) The model identification for angle of attack and pitch rate on top, the overall model error in middle and the angle of attack and pitch rate errors in bottom for BC1

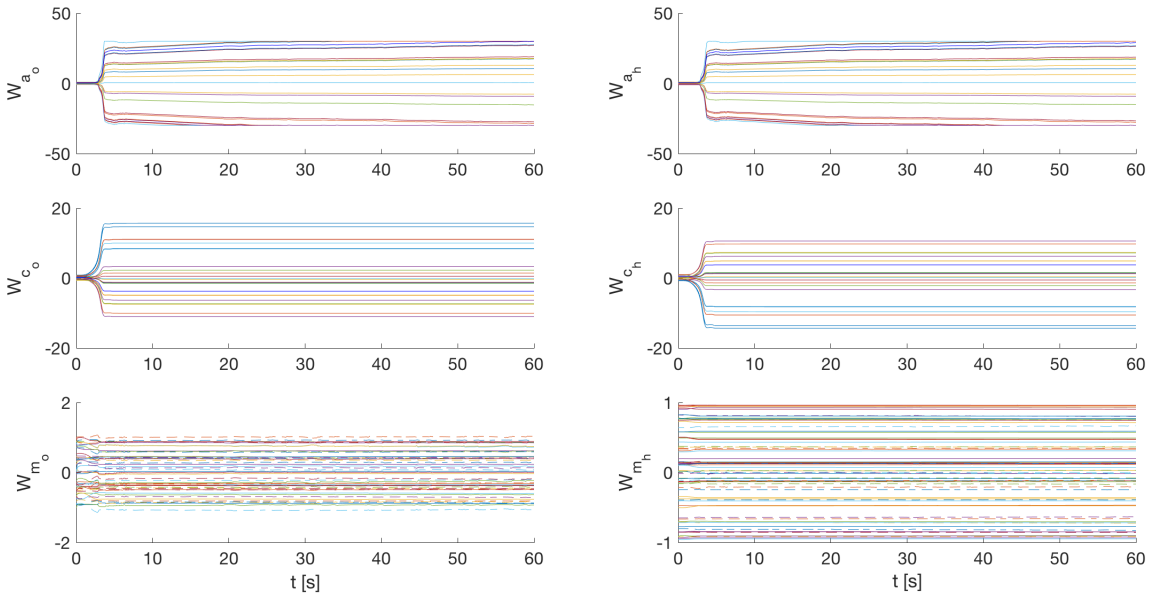


Figure (A.3) The weights update for the critic, actor and model for BC1

BC3 Results

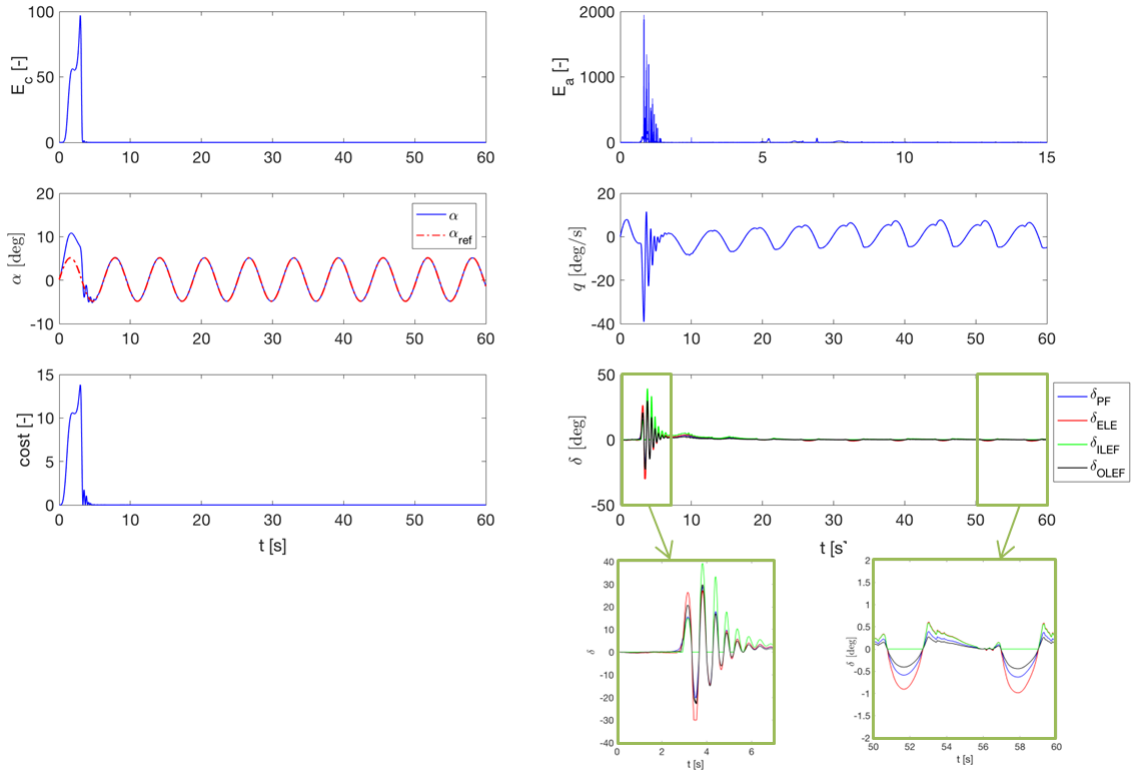


Figure (A.4) The attitude control results for BC3 case with critic network error in top left, the actor network error in top right, the angle of attack response and the reference angle of attack in middle left, the pitch rate response in middle right, the cost value in bottom left, the effectors deflection in bottom right

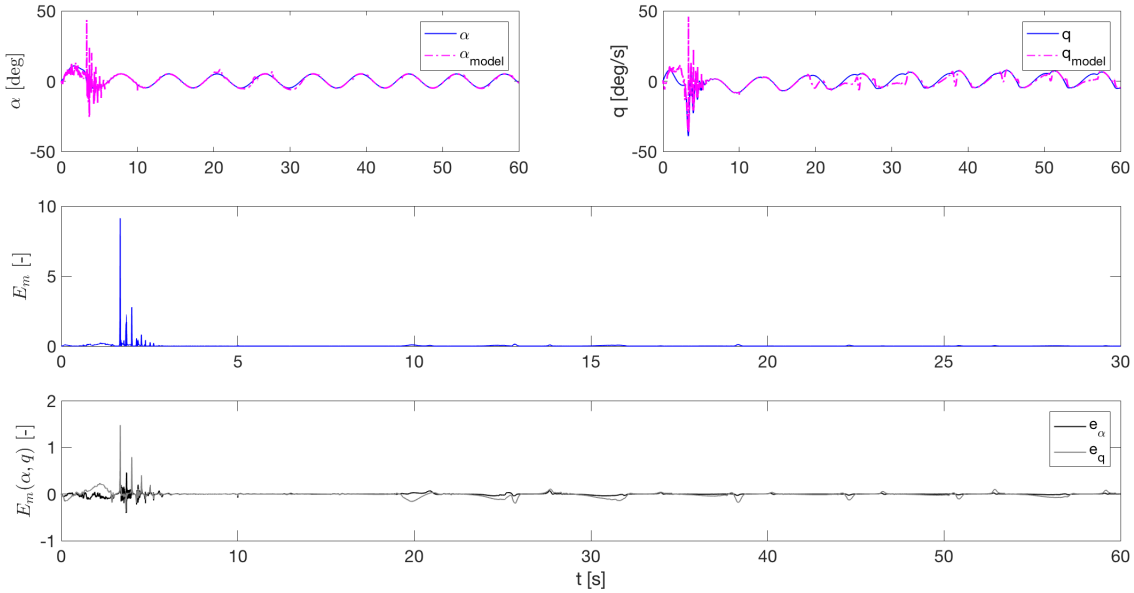


Figure (A.5) The model identification for angle of attack and pitch rate on top, the overall model error in middle and the angle of attack and pitch rate errors in bottom for BC3

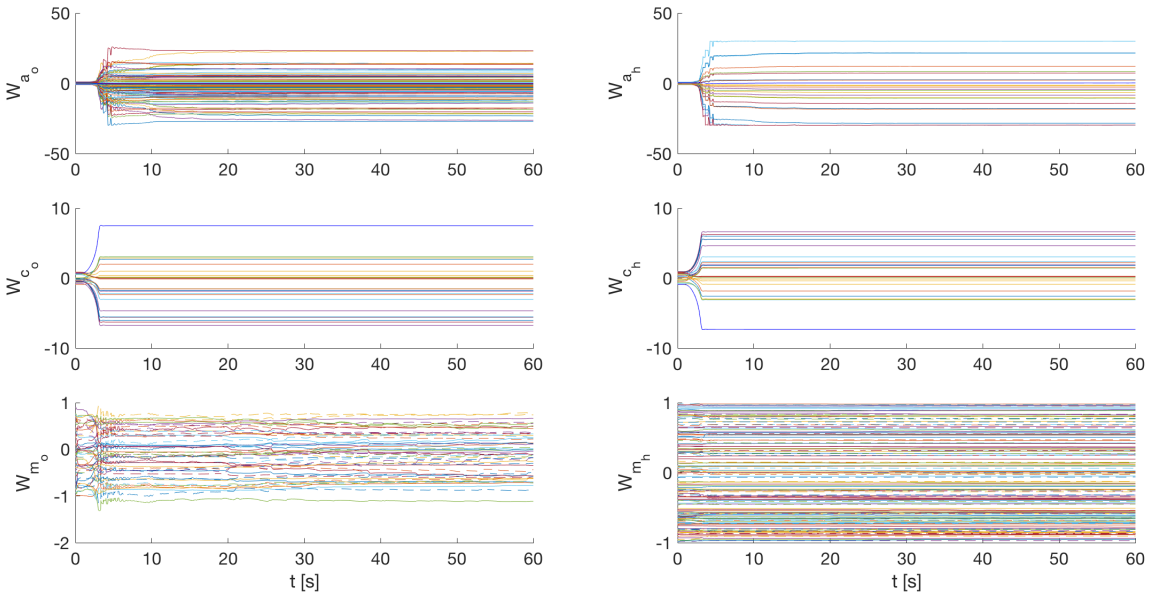


Figure (A.6) The weights update for the critic, actor and model for BC3

BC4 Results

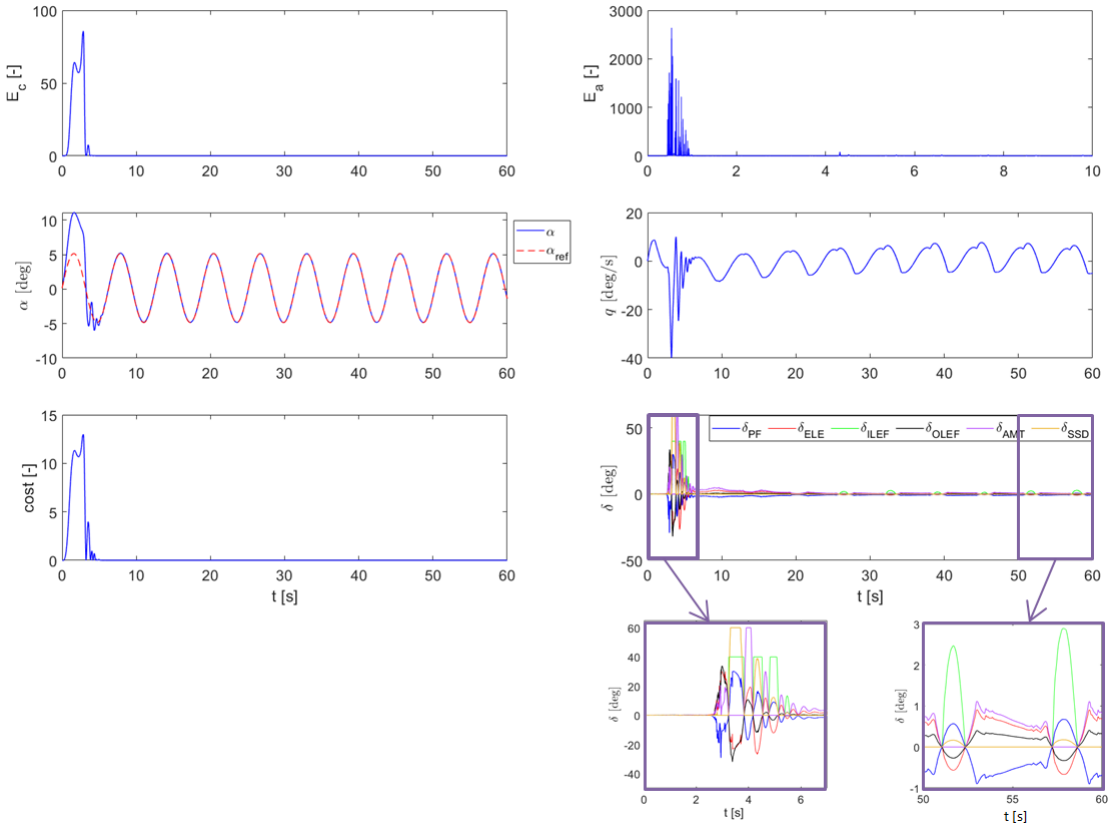


Figure (A.7) The attitude control results for BC4 case with critic network error in top left, the actor network error in top right, the angle of attack response and the reference angle of attack in middle left, the pitch rate response in middle right, the cost value in bottom left, the effectors deflection in bottom right

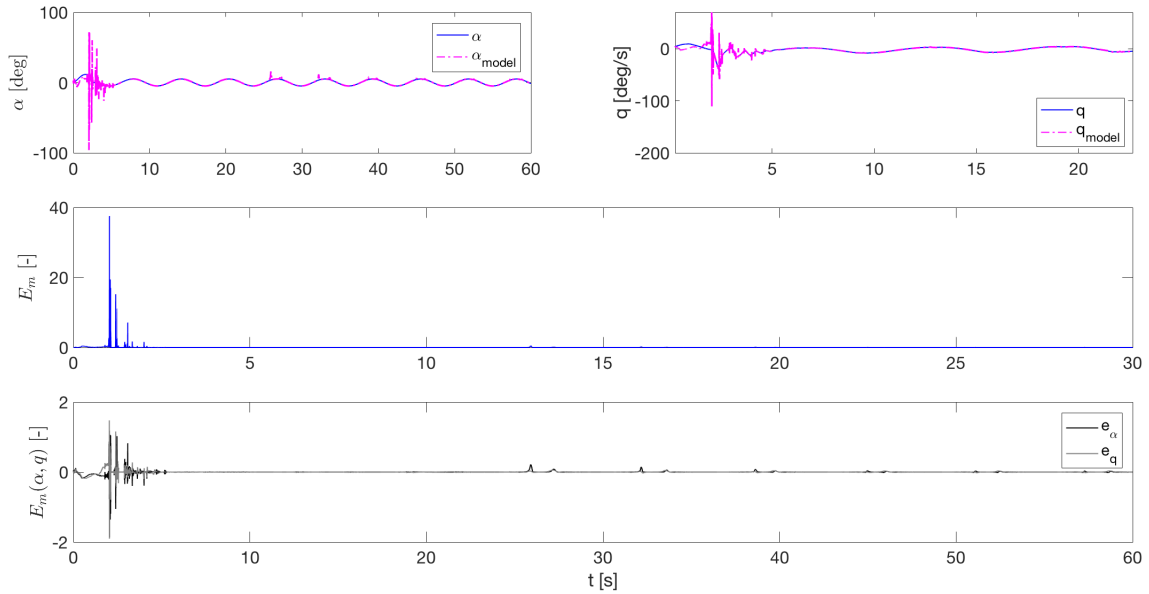


Figure (A.8) The model identification for angle of attack and pitch rate on top, the overall model error in middle and the angle of attack and pitch rate errors in bottom for BC4

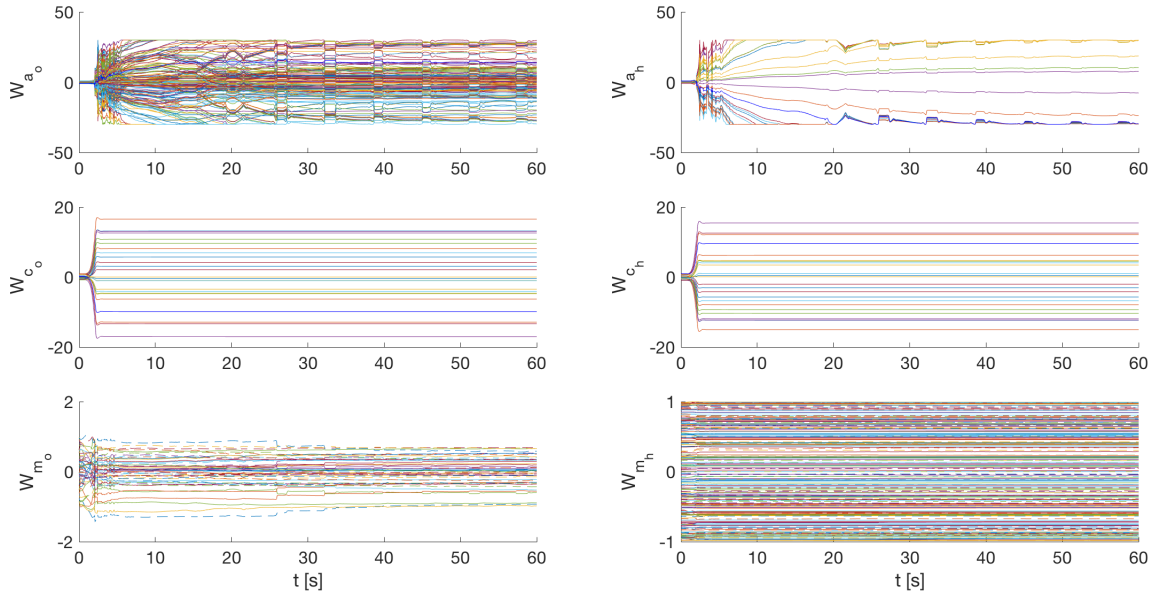


Figure (A.9) The weights update for the critic, actor and model for BC4