Reinforcement Learning for Flight Control

Learning to Fly the PH-LAB

S. Heyer

TU Delft

Reinforcement Learning for Flight Control

Learning to Fly the PH-LAB

by

S. Heyer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday March 27, 2019 at 10:00 AM.

Student number:	4151712	
Project duration:	April 3, 2018 – March 27, 2019	
Thesis committee:	Dr. ir. Q. P. Chu,	TU Delft, chair
	Dr. ir. E. van Kampen,	TU Delft, supervisor
	Dr. ir. E. Mooij,	TU Delft

TU Delft

Preface

Because of advances in machine learning, including reinforcement learning, artificial intelligence is making tremendous progress. This research field will play central roles in both future technologies and society. Therefore it is important that knowledge is accessible and evenly distributed. If you are an aspiring reinforcement learning enthusiast, you have probably already figured that it is hard and it doesn't always work. Reproducibility is a challenge, even if you have a recipe to follow and especially if you are starting from scratch, the learning curve is steep. In an effort to inspire people to work with reinforcement learning and incentivize newcomers to dig in, the algorithms developed in this thesis are made publicly available [1]. Furthermore, it should serve the reproducibility of this work, as well as, encourage others to improve upon it.

This master thesis is the result of my studies in the Control and Simulation (C&S) division at the Faculty of Aerospace Engineering at the Delft University of Technology. At the completion of this thesis and also of my student life, I would like to thank several people that supported me and shaped my path along the way. First of all, I would like to thank my supervisor Dr. ir. Erik-Jan van Kampen for his guidance and supervision. Despite the large number of students under your supervision, you never fell short of providing me with your time, patience, and attention when I needed it. Important outcomes of this thesis are a result of your valuable and constructive suggestions. To Prof. dr. ir. Max Mulder I would like to express my sincere admiration and gratitude. Your passion and investment during lectures has sparked my curiosity for many subjects and inspired me to conduct my studies with dedication. To my closest friends, I want to express my appreciation for sharing this journey with me. Thank you for all the laughs and welcome distractions as well as your help in times of frustration. I want to thank my family in Brazil and Germany for always rooting for me and gifting me with their love despite my unintentional absence. I really miss you guys. Last but certainly not least, I owe my deepest gratitude to my parents Geisa Maria Heyer and Thomas Heyer for their unconditional love and support. This one is for you.

S. Heyer
Delft, March 2019

[1] https://github.com/stefanheyer/msc-thesis

”If I have seen further it is by standing on the shoulders of giants.”

by Isaac Newton

Contents

List of Figures ix

List of Tables xi

List of Algorithms xiii

Nomenclature xv

Acronyms xix

1 Introduction 1
1.1 Research Objectives and Questions . . . 2
1.2 Outline . . . 3

I Scientific Paper 5

II Literature Survey and Preliminary Analysis 27

2 Literature Survey 29
2.1 Reinforcement Learning Fundamentals . . . 29
2.1.1 Key Concepts . . . 29
2.1.2 Common Approaches . . . 31
2.1.3 Dimensions . . . 33
2.2 Reinforcement Learning in Continuous Spaces . . . 34
2.2.1 Function Approximation . . . 34
2.2.2 Methodologies for Solving Continuous MDP . . . 35
2.2.3 Synopsis . . . 37
2.3 Approximate Dynamic Programming . . . 37
2.3.1 Adaptive Critic Designs . . . 37
2.3.2 Adaptive Critic Designs for Flight Control . . . 39
2.4 Deep Reinforcement Learning Actor-Critic Algorithms . . . 41
2.4.1 Deterministic Policy Gradient Algorithms . . . 41
2.4.2 Continuous Control with Deep Reinforcement Learning . . . 42
2.4.3 Asynchronous Methods for Deep Reinforcement Learning . . . 42
2.4.4 Policy Optimization . . . 43
2.5 Conclusion . . . 43

3 Preliminary Analysis 45
3.1 Problem Formulation . . . 45
3.1.1 Plant . . . 45
3.1.2 Flight Control Task . . . 46
3.1.3 Reward Function . . . 47
3.2 MDDHP Agent . . . 47
3.2.1 Memory Structures . . . 47
3.2.2 Update Rules . . . 49
3.2.3 Training Strategy . . . 51
3.3 IDHP Agent . . . 52
3.3.1 Incremental Model Estimation with RLS . . . 52
3.3.2 Training Strategy . . . 54
3.4 Results and Discussion . . . 54
3.4.1 Fundamental Testing . . . 56
3.4.2 Advanced Testing . . . 62

3.5 Conclusion 64

III Additional Results and Discussions **65**

4 Simulation of Online Operation in Different Flight Regimes **67**

5 Simulation of Online Operation of a Large-Angle Maneuver **73**

6 Online System Identification Model Prediction Error **75**

7 Estimator Covariance Windup **79**

8 Verification and Validation **83**

IV Closure **85**

9 Conclusion **87**

10 Recommendations **91**

V Appendices **93**

A Symmetric Equations of Motion **95**

Bibliography **97**

List of Figures

1.1 Rendering of the Cessna 550 Citation II research aircraft. 2

2.1 Agent-environment interaction in the fundamental framework of RL. 29
2.2 Representation of two main dimensions of RL methods [100]. The depth and width of the updates. 33
2.3 Selection tree of methods for function approximation and their optimization. 35
2.4 Venn diagram of Reinforcement Learning elements that are approximated. 36
2.5 DNDP-based helicopter controller integrating three cascaded artificial neural networks in one action network, a trim network, and a critic network, as presented in [23]. 40

3.1 Artificial Neural Network topology of actor (left) and critic (right). 48
3.2 Overview of Gradient Descent optimization algorithms. 49
3.3 Schematic of MDDHP framework. 50
3.4 Back-propagation schematic for the critic update in the framework of MDDHP. 51
3.5 Back-propagation schematic for the actor update in the framework of MDDHP. 52
3.6 Pitch rate reference tracking on simplified PH-LAB model with MDDHP agent. 57
3.7 Memory structure parameters of actor and critic for MDDHP agent during pitch rate reference tracking on simplified PH-LAB model. 58
3.8 Pitch rate reference tracking on simplified PH-LAB model with IDHP agent. 59
3.9 Memory structure parameters of actor and critic for IDHP agent during pitch rate reference tracking on simplified PH-LAB model. 60
3.10 Absolute estimation error of state and input matrices of incremental model with RLS identification. 60
3.11 Pitch rate reference tracking on simplified PH-LAB model comparison between IDHP and MDDHP. 61
3.12 Mean and min-max bound of 500 runs of IDHP pitch rate reference tracking on simplified PH-LAB model with untrimmed, uniform, random state initialization. 62
3.13 Pitch rate reference tracking on simplified PH-LAB model comparison between clean and noisy state measurements. 63
3.14 Pitch rate reference tracking on simplified PH-LAB model with IDHP agent in presence of sudden change to the plant's input matrix. 64

4.1 Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC0 $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines. 68
4.2 Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC2 $(V_{tas}, H) = (90\ \frac{m}{s}, 5\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines. 69
4.3 Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC1 $(V_{tas}, H) = (140\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines. 70
4.4 Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC3 $(V_{tas}, H) = (140\ \frac{m}{s}, 5\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines. 71

5.1 Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines. 74

6.1 State and input matrix estimates, and actor and critic parameters during online training of longitudinal (left) and lateral (right) controller, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$ and $(\hat{\mathbf{F}}_0, \hat{\mathbf{G}}_0, \hat{\mathbf{\Theta}}_0) = (I, \mathbf{0}, I \cdot 10^8)$. Estimates derived from a plant linearization are represented by dotted lines. 76
6.2 Absolute state increment prediction error of longitudinal (left) and lateral (right) controller and linear model, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$ and $(\hat{\mathbf{F}}_0, \hat{\mathbf{G}}_0, \hat{\mathbf{\Theta}}_0) = (I, \mathbf{0}, I \cdot 10^8)$. The prediction error of the incremental model using parameters from a plant linearization and from the online identification procedure are illustrated by red and blue line, respectively. 77
6.3 Absolute state increment prediction error percentage of longitudinal (left) and lateral (right) controller and linear model, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$ and $(\hat{\mathbf{F}}_0, \hat{\mathbf{G}}_0, \hat{\mathbf{\Theta}}_0) = (I, \mathbf{0}, I \cdot 10^8)$. The prediction error of the incremental model using parameters from a plant linearization and from the online identification procedure are illustrated by red and blue line, respectively. 77

7.1 Online operation phase, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines. A forgetting factor of $\kappa = 0.99$ is utilized for the online identification of both the longitudinal and lateral incremental models. 80
7.2 State and input matrix estimates, and variance during operation phase of longitudinal (left) and lateral (right) controller, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. Estimates derived from a plant linearization are represented by dotted lines. A forgetting factor of $\kappa = 0.99$ is utilized. 81

List of Tables

2.1 Summary of distinct attributes of different Adaptive Critic Designs structures. 38

3.1 State matrix, input matrix and sample time for short period model of PH-LAB research aircraft [58]. 46
3.2 Amplitude and frequency of sinusoidal pitch rate reference signal. 47
3.3 Agent parameters, memory structure initialization and initial state. 56
3.4 Forgetting factor, initial parameter matrix and initial covariance matrix of IDHP agent. . . 57

4.1 Description and sample failure rate of different flight conditions each simulated 100 times during the online operation phase. For all cases the agent is initialized at its online pretrained state conducted at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. 67

A.1 Symbols as utilized in Equation (A.1) [58]. 96
A.2 Symmetric stability and control derivatives [58]. 96

List of Algorithms

3.1 Model Dependent Dual Heuristic Programming 53
3.2 Incremental Dual Heuristic Programming 55

Nomenclature

$(a, b]$	the real interval between a and b including b but not including a
$\approx$	approximately equal
$\mathbf{I}$	identity matrix
$\doteq$	equality relationship that is true by definition
$\mathbb{1}_{predicate}$	indicator function ($\mathbb{1}_{predicate} \doteq 1$ if the *predicate* is true, else 0)
$\mathbb{R}$	set of real numbers
$\leftarrow$	assignment
$\mathbb{E}[X]$	expectation of a random variable X
$\mathcal{N}_{trunc}$	truncated Gaussian distribution
$\Pr\{X = x\}$	probability that a random variable X takes on the value x
$\mathcal{A}$	set of all actions
$\mathcal{R}$	set of all rewards
$\mathcal{S}$	set of all states
Δt	sample time
T	simulation duration
t	time
A	amplitude
f	frequency
a	action
g	return
r	reward
s, s'	states
A_t	action at time t as random variable
G_t	return following time t as random variable

R_t	reward at time t as random variable
S_t	state at time t as random variable
$\mathbf{a}$	action vector
$\mathbf{s}$	state vector
$\mathbf{s}^R$	reference state vector
$\Delta\mathbf{a}$	action vector increment
$\Delta\mathbf{s}$	state vector increment
$\Delta\hat{\mathbf{s}}$	state vector rate of change
$\dot{\mathbf{s}}$	rate of change of state vector
$f(\mathbf{s}, \mathbf{a})$	state-transition function
$p(s'\|s, a)$	probability of transition to state s', from state s and action a
$p(s', r\|s, a)$	probability of transition to state s' with reward r, from state s and action a
$r(\mathbf{s}^R, \mathbf{s})$	immediate reward as function of state vector and reference state vector
$r(s, a)$	expected immediate reward on transition from s under action a
$r(s, a, s')$	expected immediate reward on transition from s to s' under action a
$\hat{\pi}(\mathbf{s}, \mathbf{s}^R, \mathbf{w}^A)$	parametric approximation of the policy
π	policy (decision-making rule)
$\pi(\mathbf{s}, \mathbf{s}^R)$	deterministic policy as function of state vector and reference state vector
$\pi(a\|s)$	probability of taking action a in state s under *stochastic* policy π
$\mathbf{e}$	partial derivative of the temporal difference error w.r.t. state vector
δ_t	temporal-difference error at t
$\hat{\lambda}(\mathbf{s}, \mathbf{s}^R, \mathbf{w}^C)$	parametric approximation of state-value partial derivative w.r.t. state vector
$\lambda(\mathbf{s}, \mathbf{s}^R)$	state-value partial derivative w.r.t. state vector
$a_\pi(s, a)$	advantage of taking action a in state s under policy π
$Q(s, a)$	estimate of action-value function q_π or q_*
$q_*(s, a)$	value of taking action a in state s under the optimal policy
$q_\pi(s, a)$	value of taking action a in state s under policy π
$v(\mathbf{s}, \mathbf{s}^R)$	state-value as function of state vector and reference state vector
$V(s)$	estimate of state-value function v_π or v_*

$v_*(s)$	value of state s under the optimal policy
$v_\pi(s)$	value of state s under policy π
$\mathbf{A}$	state matrix for continuous system
$\mathbf{B}$	input matrix for continuous system
$\mathbf{F}$	state matrix for discrete-time system
$\mathbf{G}$	input matrix for discrete-time system
$\hat{\mathbf{F}}$	state matrix for discrete-time system estimate
$\hat{\mathbf{G}}$	input matrix for discrete-time system estimate
$\mathbf{P}$	Boolean selection matrix
$\mathbf{Q}$	weight matrix
$\mathbf{\Lambda}$	covariance matrix
$\mathbf{X}$	measurement matrix
ϵ	innovation term
$\hat{\mathbf{\Theta}}$	parameter matrix
L	loss
L^A	loss actor
L^C	loss critic
$\mathbf{w}, \mathbf{w}'$	parameter vector
$\mathbf{w}^A$	parameter vector actor
$\mathbf{w}^C$	parameter vector critic
α	step-size parameter
η	learning rate
η^A	learning rate actor
η^C	learning rate critic
γ	discount factor
κ	forgetting factor
τ	mixing factor

α	angle of attack
β	sideslip angle
$\Delta\alpha$	angle of attack increment
$\Delta\beta$	sideslip angle increment
$\Delta\hat{\alpha}$	angle of attack increment prediction
$\Delta\hat{\beta}$	sideslip angle increment prediction
$\Delta\hat{\phi}$	roll angle increment prediction
$\Delta\hat{\theta}$	pitch angle increment prediction
$\Delta\hat{p}$	roll rate increment prediction
$\Delta\hat{q}$	pitch rate increment prediction
$\Delta\hat{r}$	yaw rate increment prediction
$\Delta\phi$	roll angle increment
$\Delta\theta$	pitch angle increment
Δp	roll rate increment
Δq	pitch rate increment
Δr	yaw rate increment
δ_a	aileron deflection
δ_e	elevator deflection
δ_r	rudder deflection
ϕ	roll angle
θ	pitch angle
H	altitude
p	roll rate
q	pitch rate
q^R	reference pitch rate
r	yaw rate
V_{tas}	true airspeed

Acronyms

A2C Advantage Actor-Critic.
A3C Asynchronous Advantage Actor-Critic.
ACDs Adaptive Critic Designs.
ACER Actor-Critic with Experience Replay.
ADDHP Action Dependent Dual Heuristic Programming.
ADGDHP Action Dependent Globalized Dual Heuristic Programming.
ADHDP Action Dependent Heuristic Dynamic Programming.
ADP Approximate Dynamic Programming.
ANN Artificial Neural Network.
ANNs Artificial Neural Networks.

BS Backstepping.

D4PG Distributed Distributional Deep Deterministic Policy Gradients.
DASMAT Delft University Aircraft Simulation Model and Analysis Tool.
DDPG Deep Deterministic Policy Gradients.
DHP Dual Heuristic Programming.
DNDP Direct Neural Dynamic Programming.
DP Dynamic Programming.
DPG Deterministic Policy Gradient.
DQN Deep Q-Network.
DRL Deep Reinforcement Learning.

FCNN Fully Connected Neural Network.

GAE Generalized Advantage Estimator.
GD Gradient Descent.
GDHP Globalized Dual Heuristic Programming.
GPI Generalized Policy Iteration.

HDP Heuristic Dynamic Programming.
HER Hindsight Experience Replay.

IBS Incremental Backstepping.
IDHP Incremental Dual Heuristic Programming.
IHDP Incremental Heuristic Dynamic Programming.
ILS Instrument Landing System.
INDI Incremental Nonlinear Dynamic Inversion.

KL Kullback–Leibler.

LQR Linear Quadratic Regulator.
LSPE Least Squares Policy Evaluation.
LSPI Least Squares Policy Iteration.

LSTD Least Squares Temporal Difference.
LSTM Long Short Term Memory.
LTI Linear Time Invariant.

MADDPG Multi-Agent Deep Deterministic Policy Gradients.
MC Monte Carlo.
MDDHP Model Dependent Dual Heuristic Programming.
MDP Markov Decision Process.
MDPs Markov Decision Processes.
MIMO Multiple Inputs Multiple Outputs.
ML Machine Learning.

NAG Nesterov Accelerated Gradient.
NDI Nonlinear Dynamic Inversion.

OLS Ordinary Least Squares.

PER Prioritized Experience Replay.
POMDPs Partially Observable Markov Decision Processes.
PPO Proximal Policy Optimization.

ReLU Rectified Linear Unit.
RL Reinforcement Learning.
RLS Recursive Least Squares.

SCN Structured Control Net.
SGD Stochastic Gradient Descent.

TD Temporal Difference.
TD3 Twin Delayed Deep Deterministic.
TRPO Trust Region Policy Optimization.

UAVs Unmanned Aerial Vehicles.

VTOL Vertical Take-Off and Landing.

1 Introduction

In recent years, the aerospace domain has experienced an unprecedented increase in interest in autonomous operations. Unmanned Aerial Vehicles (UAVs), originally targeted to consumer-focused recreation have proven their value for business and are today an indispensable tool in a variety of industries, such as construction [28], agriculture [11] and mapping [62]. A similar process is occurring in the general aerial transportation sector. Major challenges in battery technology, electric and distributed propulsion, and Vertical Take-Off and Landing (VTOL) are being solved with innovation in aircraft design, enabling a visionary future with electric, autonomous flying-cars crossing the skies in large urban areas. The idea of flying-cars is transitioning from a fascinating concept to a mature technological solution for sustainable transportation and rapid urbanization, disrupting nothing less than the entire mobility industry [50]. As a result, several start-ups, such as Lilium, Volocopter, Kittyhawk, backed by large corporations, such as Google and Daimler AG, are competing with established aerospace companies. Nevertheless, this new business model requires a fully autonomous operation to be economically viable. Autonomous systems, especially when operated in complex urban environments, need to be able to adapt to sudden, unexpected changes in the environment [118] and to changes in their dynamic system, also referred to as fault tolerance. Additionally to the requirement of online operation, model-independence is important, as for many aerospace applications no accurate system model is available, nor readily identifiable [83][51].

Current flight control systems for passenger aircraft predominantly rely on classical control techniques, which make use of multiple linear controllers. Gain scheduling methods are then applied to switch between the numerous linear controllers, each designed for a specific operating point [7]. These gains are determined offline, in advance, by means of a model of the system. Their non-adaptive nature makes them unsuitable for autonomous systems. Since the 1960s, adaptive control has been an active research field, developing control strategies that adapt to changing system conditions online. Many, such as Nonlinear Dynamic Inversion (NDI) [44] [16] and Backstepping (BS) [94], with the focus on dealing with system nonlinearities. Although successfully applied [24] [92] [93] [95] [108], these methods strongly rely on an accurate model of the system dynamics. Recently developed incremental control methods, such as Incremental Nonlinear Dynamic Inversion (INDI), Incremental Backstepping (IBS) and incremental adaptive sliding mode control have decreased the model-dependence, in exchange for the need of high sample rate measurements [53][87][90][1][91][2]. Furthermore, major steps have been made in INDI through the first stability and robustness analysis in the presence of external disturbances [112] and a first successful demonstration on an CS-25 certified aircraft [33].

Originally inspired by the idea of replicating biological learning mechanisms [82], Reinforcement Learning (RL) is a field of Machine Learning (ML) that is best characterized by learning from interaction [100]. Ever since RL has been studied and applied to the field of adaptive and optimal control. Traditional RL methods were formulated for discrete state and actions spaces, which were sufficiently small such that approximate value functions could be represented as tables. Continuous and high-dimensional spaces prevalent in control applications would lead to an exponential growth in computational complexity known as the curse of dimensionality [70]. The curse of dimensionality was mitigated with the introduction of function approximators, such as Artificial Neural Networks (ANNs), characterizing the field of Approximate Dynamic Programming (ADP) [86]. With Adaptive Critic Designs (ACDs),

a class of ADP methods, several applications were successfully explored in the 2000s, including adaptive flight control for a missile system [12], business jet [27], helicopter [23] and military aircraft [107]. However, these methods often need an extra structure to approximate the system dynamics. Furthermore, when applied online, a preceding offline learning phase is required, mainly due to non-trivial identification of the system dynamics. The offline identification phase itself requires a representative simulation model.

Progress in RL research has accelerated significantly in recent years, due to the advent of deep learning, establishing the field of Deep Reinforcement Learning (DRL). Major contributions such as Deep Q-Network (DQN) [56] and AlphaGo [89], have demonstrated solutions to previously intractable decision-making problems. An overview of recent efforts in DRL is presented in [5]. At the same time, progress in ADP research has proposed novel frameworks. Most interesting are new ACDs, such as Incremental Heuristic Dynamic Programming (IHDP) [123] and Incremental Dual Heuristic Programming (IDHP) [124], that utilize the incremental model technique, first introduced for INDI and IBS. IHDP and IDHP are able to improve online adaptability and most importantly, eliminate the need for an offline learning phase, by identifying an incremental model of the system in real-time. However, these recent advances have yet to be applied to and validated on complex, high-dimensional aerospace models and real systems.

The proposed thesis research aims to advance the current state-of-the-art in adaptive flight control, by demonstrating an implementation of recent advances in RL research to complex, high-dimensional aerospace systems. This is achieved through the derivation, design, and analysis of an online, model-independent, adaptive flight controller for the PH-LAB research aircraft, a Cessna 550 Citation II, depicted in Figure 1.1.

Figure 1.1: Rendering of the Cessna 550 Citation II research aircraft.

1.1. Research Objectives and Questions

The aim of the research is captured by the formulation of the main goal **MG**. The main goal is accomplished by concluding the more explicit sub-goals (G1, G2, G3, G4, G5, G6). These constitute the main building blocks to be sequentially achieved during the proposed research to attain the main objective.

The main goal and accompanying sub-goals are defined as followed.

MG Contribute to the development of *online*, *model-independent*, *adaptive* flight control for fixed wing aircraft with the purpose of dealing with *complex*, *continuous* dynamical systems, **by** *investigating* the applicability of novel Reinforcement Learning frameworks, through the *design* of a flight controller for the PH-LAB research aircraft.

(G1) Identify a RL framework candidate, through the review of state-of-the-art RL algorithms and the identification of their limitations.

(G2) Reproduce the proposed RL framework and conduct a verification by comparing with published empirical results.

(G3) Integrate the proposed RL framework to the PH-LAB research aircraft model, through the development of an interface.

(G4) Develop a baseline tracking flight controller, through an incremental increase of task complexity.

(G5) Propose modifications to the RL framework to improve adaptability and learning stability of the baseline controller.

(G6) Evaluate the controller's ability to operate the aircraft online.

Subsequently, the research questions are formulated based on the set of research goals. The research questions **RQ1**, **RQ2**, **RQ3**, and **RQ4** when answered, generate the insights that are key to achieving the objectives. The sub-questions then break down the main questions into specific and measurable queries that can be tested and answered empirically.

The main questions to be answered through the evaluation of the sub-questions are as followed.

RQ1 Which RL algorithm is most applicable to the adaptive flight control of the PH-LAB research aircraft?

(RQ1.1) What are the current state-of-the-art RL algorithms for continuous control?

(RQ1.2) Is the proposed RL framework reproducible?

(RQ1.3) Does the proposed RL framework demonstrate satisfactory performance when applied to a simplified version of the PH-LAB research aircraft?

RQ2 How can the proposed RL framework be incorporated into the flight controller design?

(RQ2.1) What are the dynamical system characteristics inherent to the PH-LAB research aircraft?

(RQ2.2) At which control level should the RL framework be embedded?

(RQ2.3) What are the implications of continual learning for the RL framework?

RQ3 How can the proposed RL framework be modified to improve controller adaptability and learning stability?

RQ4 What is the overall performance of the RL flight controller design?

(RQ4.1) What is the performance of the learning process measured in stability and speed?

(RQ4.2) How does it generalize to different flight regimes?

(RQ4.3) How does it perform during unexpected changes to the system or framework?

1.2. Outline

The remainder of this thesis is structured as follows. In Part I the conducted research is presented as a scientific paper. First, the paper elaborates on the foundations by defining the problem statement and presenting the derivation of the proposed learning framework. Subsequently, the paper discusses how the learning framework is embedded in the controller design and introduces the high-fidelity simulation model utilized in this research. This is followed by the presentation and discussion of the results of the conducted experiments, which focus on online training, online operation and online adaption. In Part II, the groundwork of the thesis research is presented [1]. A literature survey is presented in Chapter 2 and includes the fundamentals of RL, an overview of the landscape of RL methods and a review and analysis of the publications in the scope of the research objective. The literature survey is concluded with the proposal of a baseline RL framework. Readers unfamiliar with the subject are advised to read Part II before engaging with the paper. In Chapter 3, a preliminary analysis of the proposed framework is conducted by applying it to a simplified model of the Cessna 550 Citation II. In Part III, additional results are presented and discussed. First, additional simulation results of the controller's performance for additional flight conditions, including large-angle maneuvers, is presented. Subsequently, an extended analysis of the performance of the online system identification of the incremental model is presented by discussing the resulting prediction errors. In addition, an example of an estimator windup event is presented and its effect on the agent's performance is discussed. Last but not least, the verification and validation processes conducted in this thesis are discussed. In Part IV, the thesis is concluded and additional recommendations for future research are provided.

[1] Part II was assessed as part of the AE4020 Literature Study course.

I

Scientific Paper

Online Adaptive Incremental Reinforcement Learning Flight Control for a CS-25 Class Aircraft

S. Heyer*
Delft University of Technology, P.O. Box 5058, 2600GB Delft, The Netherlands

In recent years Adaptive Critic Designs (ACDs) have been applied to adaptive flight control of uncertain, nonlinear systems. However, these algorithms often rely on representative models as they require an offline training stage. Therefore, they have limited applicability to a system for which no accurate system model is available, nor readily identifiable. Inspired by recent work on Incremental Dual Heuristic Programming (IDHP), this paper derives and analyzes a Reinforcement Learning (RL) based framework for adaptive flight control of a CS-25 class fixed-wing aircraft. The proposed framework utilizes Artificial Neural Networks (ANNs) and includes an additional network structure to improve learning stability. The designed learning controller is implemented to control a high-fidelity, six-degree-of-freedom simulation of the Cessna 550 Citation II PH-LAB research aircraft. It is demonstrated that the proposed framework is able to learn a near-optimal control policy online without a priori knowledge of the system dynamics nor an offline training phase. Furthermore, it is able to generalize and operate the aircraft in not previously encountered flight regimes as well as identify and adapt to unforeseen changes to the aircraft's dynamics.

Nomenclature

$\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{a}$	=	state, reference state, and action vectors
r, g	=	reward and return
γ, τ, κ	=	discount, mixing, and forgetting factors
$f(\boldsymbol{s}, \boldsymbol{a}), r(\boldsymbol{s}^R, \boldsymbol{s})$	=	state-transition and reward functions
$\pi(\boldsymbol{s}, \boldsymbol{s}^R), \pi^*(\boldsymbol{s}, \boldsymbol{s}^R)$	=	policy and optimal policy
$\hat{\pi}(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^A)$	=	parametric approximation of the policy
$v(\boldsymbol{s}, \boldsymbol{s}^R)$	=	state-value function
$\lambda(\boldsymbol{s}, \boldsymbol{s}^R)$	=	state-value partial derivative w.r.t. the state vector
$\hat{\lambda}(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^C)$	=	parametric approximation of state-value partial derivative w.r.t. state vector
$\hat{\lambda}'(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^{C'})$	=	parametric approximation of state-value partial derivative w.r.t. state vector
$\boldsymbol{P}, \boldsymbol{Q}$	=	Boolean selection and symmetric weight matrices
$t, \Delta t, T$	=	time, sample time, and simulation time duration
$\boldsymbol{F}, \hat{\boldsymbol{F}}$	=	state matrix and state matrix estimate
$\boldsymbol{G}, \hat{\boldsymbol{G}}$	=	input matrix and input matrix estimate
$\boldsymbol{w}^A, \boldsymbol{w}^C, \boldsymbol{w}^{C'}$	=	actor, critic, and target critic parameter vectors
$\Delta\boldsymbol{s}, \Delta\boldsymbol{a}, \Delta\hat{\boldsymbol{s}}$	=	state and action vector increment and state vector increment prediction
$\boldsymbol{e}$	=	partial derivative of the Temporal Difference (TD) error w.r.t. the state vector
L^A, L^C	=	actor and critic losses
η^A, η^C	=	actor and critic learning rates
$\hat{\boldsymbol{\Theta}}, \boldsymbol{\Lambda}, \boldsymbol{X}$	=	Recursive Least Squares (RLS) parameter, covariance, and measurement matrices
$\boldsymbol{\epsilon}$	=	innovation vector
$\delta_e, \delta_a, \delta_r$	=	elevator, aileron, and rudder deflections
$p, q, r, \alpha, \beta, \phi, \theta$	=	roll rate, pitch rate, yaw rate, angle of attack, sideslip angle, roll angle, and pitch angle
$p^R, q^R, \beta^R, \phi^R, H^R, \gamma^R$	=	roll rate, pitch rate, sideslip angle, roll angle, altitude, and flight path angle reference values
V_{tas}, H, γ, n	=	true airspeed, altitude, flight path angle, and load factor

*Graduate Student, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology

I. Introduction

In recent years, the aerospace domain has experienced an unprecedented increase in interest in autonomous operations. Autonomous systems, especially when operated in complex urban environments, need to be able to adapt to sudden, unexpected changes in the environment [1] and to changes in their dynamics, also referred to as fault tolerance. Additionally to the requirement of online operation, model-independence is important as for many aerospace applications, no accurate system model is available, nor readily identifiable [2, 3].

Current flight control systems for passenger aircraft predominantly rely on classical control techniques, which make use of multiple linear controllers. Gain scheduling methods are then applied to switch between the numerous linear controllers, each designed for a specific operating point [4]. These gains are determined offline, in advance, by means of a model of the system. Their non-adaptive nature makes them unsuitable for autonomous systems.

Since the 1960s, adaptive control has been an active research field, developing control strategies that adapt to changing system conditions online. Many, such as Nonlinear Dynamic Inversion (NDI) [5, 6] and Backstepping (BS) [7], with the focus on dealing with system nonlinearities. Although successfully applied [8–12], these methods strongly rely on an accurate model of the system dynamics. Recently developed incremental control methods, such as Incremental Nonlinear Dynamic Inversion (INDI), Incremental Backstepping (IBS) and incremental adaptive sliding mode control have decreased the model-dependence, in exchange for the need of high sample rate measurements [13–18]. Furthermore, major steps have been made in INDI through the first stability and robustness analysis in the presence of external disturbances [19] and a first successful demonstration on an CS-25 certified aircraft [20].

Originally inspired by the idea of replicating biological learning mechanisms [21], Reinforcement Learning (RL) is a field of Machine Learning (ML) that is best characterized by learning from interaction [22]. Ever since RL has been studied and applied to the field of adaptive and optimal control. Traditional RL methods were formulated for discrete state and actions spaces, which were sufficiently small such that approximate value functions could be represented as tables. Continuous and high-dimensional spaces prevalent in control applications would lead to an exponential growth in computational complexity known as the curse of dimensionality [23]. The curse of dimensionality was mitigated with the introduction of function approximators, such as Artificial Neural Networks (ANNs), characterizing the field of Approximate Dynamic Programming (ADP) [24].

With Adaptive Critic Designs (ACDs), a class of ADP methods, several applications were successfully explored in the 2000s, including adaptive flight control for a missile system [25], business jet [26], helicopter [27] and military aircraft [28]. However, these methods often need an extra structure to approximate the system dynamics. Furthermore, when applied online, a preceding offline learning phase is required, mainly due to non-trivial identification of the system dynamics. The offline identification phase itself requires a representative simulation model. In [29, 30] novel frameworks, named Incremental Heuristic Dynamic Programming (IHDP) and Incremental Dual Heuristic Programming (IDHP) have been proposed to improve online adaptability and most importantly, eliminate the current need of an offline learning phase, by identifying an incremental model of the system in real-time. However, these novel frameworks have yet to be applied to and validated on complex, high-dimensional aerospace models and real systems.

The main contribution of this paper is to present the design and analysis of a RL based adaptive flight controller for a CS-25 class fixed-wing research aircraft, that can learn a near-optimal control policy online without a priori knowledge of the system dynamics nor an offline training phase. In this work, a novel learning algorithm, based on IDHP, is proposed and applied to a six-degree-of-freedom, high-fidelity, nonlinear simulation model of a Cessna 550 Citation II PH-LAB research aircraft. Through simulation, it is shown that the designed controller, is able to learn to control the aircraft, without a priori system knowledge, during a short online training phase. However, learning instability inherent to IDHP can lead to failures, which are unacceptable during operation in real systems. Therefore the proposed algorithm utilizes a separate target critic network to improve learning stability during Temporal Difference (TD) backups. Furthermore, it is shown that the controller is able to generalize and operate the aircraft in not previously encountered flight regimes and identify and adapt to unforeseen changes to the aircraft's dynamics.

The remainder of this paper is structured as follows. In Section II the control problem is formulated and the proposed learning framework is derived. Section III introduces the high-fidelity simulation model and presents the implementation of the learning framework into the controller design. Subsequently, in Section IV, the controller is tested and evaluated in three distinct cases: (1) online training, (2) online operation and (3) online adaption. Lastly, this paper is concluded in Section V.

Fig. 1 CS-25 class Cessna 550 Citation II PH-LAB research aircraft operated by the Delft University of Technology.

II. Foundations

This section starts by formulating the flight control task as a RL problem. Subsequently, the proposed learning framework is introduced, followed by its update rules and training strategy.

A. Problem Formulation

In the framework of RL the state-transition and reward function are commonly defined as processes of the environment, whose mechanics are hidden from the agent [22]. This paper regards them as separate entities, where the reward function is not a hidden process of the environment, but a designed functional. The state-transition function $f(\boldsymbol{s}, \boldsymbol{a}) \in \mathbb{R}^{m\times 1}$ characterizes the discrete-time, deterministic, nonlinear plant as in Eq. (1) with the state vector $\boldsymbol{s} \in \mathbb{R}^{m\times 1}$, the action vector $\boldsymbol{a} \in \mathbb{R}^{n\times 1}$ and the assumption of synchronous, high-frequency sampling.

The flight control problem is a variable set-point tracking task with the reference state vector $\boldsymbol{s}^R \in \mathbb{R}^{p\times 1}$. Therefore, the goal is to learn a deterministic policy $\pi(\boldsymbol{s}, \boldsymbol{s}^R) \in \mathbb{R}^{n\times 1}$ that maximizes the scalar return g, defined by Eq. (2) and Eq. (3), respectively. The return, defined by the state-value function $v(\boldsymbol{s}, \boldsymbol{s}^R)$, represents a discounted sum of future scalar rewards r, where the scalar discount factor $\gamma \in [0, 1]$ is a property of the agent. Equation (4) is the reward function. It is defined as the negative, weighted, squared state tracking error, with the Boolean selection matrix $\boldsymbol{P} \in \mathbb{R}^{p\times m}$ and the symmetric weight matrix $\boldsymbol{Q} \in \mathbb{R}^{p\times p}$. Furthermore, the reward function is differentiable as required for the update operations of Dual Heuristic Programming (DHP) frameworks. Its partial derivative with respect to the state vector $\frac{\partial r}{\partial \boldsymbol{s}} \in \mathbb{R}^{1\times m}$ is defined in Eq. (5).

$$\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t) \tag{1}$$

$$\boldsymbol{a}_t = \pi(\boldsymbol{s}_t, \boldsymbol{s}_t^R) \tag{2}$$

$$g_t = v(\boldsymbol{s}_t, \boldsymbol{s}_t^R) = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \tag{3}$$

$$r_{t+1} = r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1}) = -\left[\boldsymbol{P}\boldsymbol{s}_{t+1} - \boldsymbol{s}_t^R\right]^T \boldsymbol{Q} \left[\boldsymbol{P}\boldsymbol{s}_{t+1} - \boldsymbol{s}_t^R\right] \tag{4}$$

$$\frac{\partial r_{t+1}}{\partial \boldsymbol{s}_{t+1}} = \frac{\partial r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1})}{\partial \boldsymbol{s}_{t+1}} = -2\left[\boldsymbol{P}\boldsymbol{s}_{t+1} - \boldsymbol{s}_t^R\right]^T \boldsymbol{Q}\boldsymbol{P} \tag{5}$$

B. Learning Framework

A schematic of the feed-forward signal flow of the proposed learning framework is presented in Fig. 2. The schematic includes both processes and parametric structures. The proposed learning framework is derived from the IDHP framework as presented in [30], with its three parametric structures: the actor $\hat{\pi}(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^A) \in \mathbb{R}^{n\times 1}$, the critic $\hat{\lambda}(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^C) \in \mathbb{R}^{1\times m}$, and the incremental model of the plant. Whereas the actor approximates the control policy, the critic approximates the partial derivative of the state-value function with respect to the state, with the parameter vectors $\boldsymbol{w}^A$ and $\boldsymbol{w}^C$, respectively.

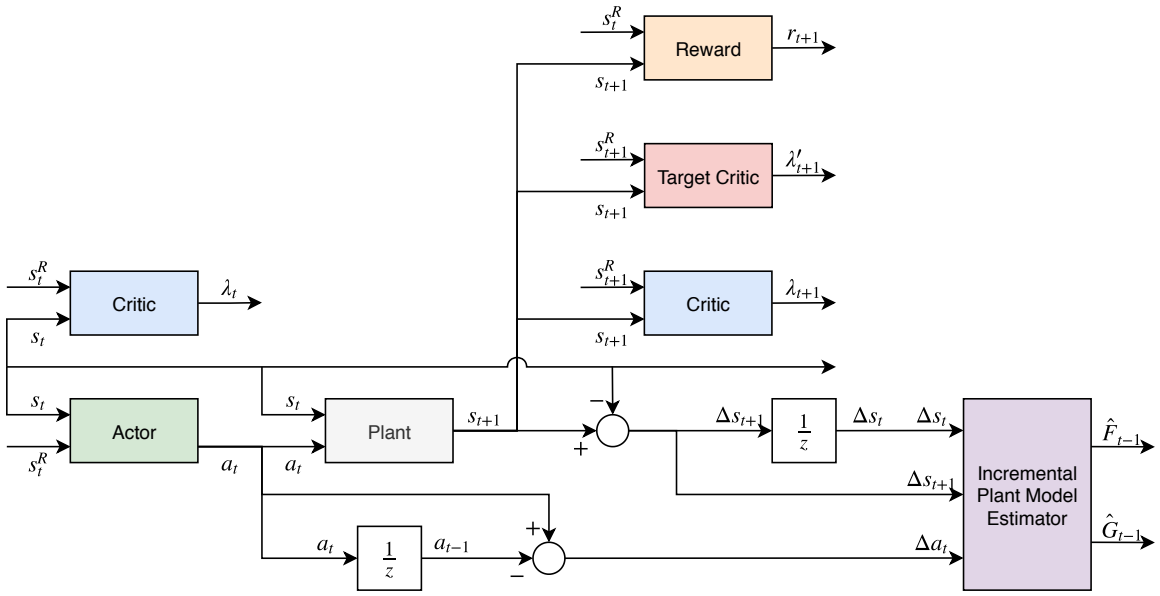

Fig. 2 Schematic of the feed-forward signal flow of the learning framework, where the different parametric structures and processes are illustrated by distinct colors.

1. Target Network

The proposed learning framework is an on-policy approach and therefore does not appertain to the deadly triad*. Nevertheless, any form of learning instability is undesirable during online operation. To improve learning stability a separate target critic $\hat{\lambda}'(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^{C'}) \in \mathbb{R}^{1\times m}$, inspired by [31, 32], is proposed for the TD backups. The proposed approach slows down learning as the propagation of the state-value derivatives is delayed, but this is outweighed by the improved learning stability. Analogous to the actor and critic, the target critic has a parameter vector $\boldsymbol{w}^{C'}$.

2. Incremental Model

By means of a first-order Taylor series expansion of the discrete-time, nonlinear plant at the operating point $[\boldsymbol{s}_0, \boldsymbol{a}_0]$ a linear approximation of the system is established as in Eq. (6), with the partial derivatives of the state-transition function, $\boldsymbol{F}(\boldsymbol{s}_0, \boldsymbol{a}_0) = \frac{\partial f(\boldsymbol{s}_0, \boldsymbol{a}_0)}{\partial \boldsymbol{s}_0} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{G}(\boldsymbol{s}_0, \boldsymbol{a}_0) = \frac{\partial f(\boldsymbol{s}_0, \boldsymbol{a}_0)}{\partial \boldsymbol{a}_0} \in \mathbb{R}^{m\times n}$, also referred to as the state matrix and input matrix, respectively. By choosing the operating point $[\boldsymbol{x}_0, \boldsymbol{a}_0] = [\boldsymbol{x}_{t-1}, \boldsymbol{a}_{t-1}]$ and rearranging Eq. (6), the incremental form of the discrete-time, linear approximation of the system is obtained, as in Eq. (7), with $\Delta\boldsymbol{s}_{t+1} = \boldsymbol{s}_{t+1} - \boldsymbol{s}_t$, $\Delta\boldsymbol{a}_{t+1} = \boldsymbol{a}_{t+1} - \boldsymbol{a}_t$, $\boldsymbol{F}_{t-1} = \boldsymbol{F}(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})$, and $\boldsymbol{G}_{t-1} = \boldsymbol{G}(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})$.

$$\boldsymbol{s}_{t+1} \approx f(\boldsymbol{s}_0, \boldsymbol{a}_0) + \boldsymbol{F}(\boldsymbol{s}_0, \boldsymbol{a}_0)[\boldsymbol{s}_t - \boldsymbol{s}_0] + \boldsymbol{G}(\boldsymbol{s}_0, \boldsymbol{a}_0)[\boldsymbol{a}_t - \boldsymbol{a}_0] \tag{6}$$

$$\Delta\boldsymbol{s}_{t+1} \approx \boldsymbol{F}_{t-1}\Delta\boldsymbol{s}_t + \boldsymbol{G}_{t-1}\Delta\boldsymbol{a}_t \tag{7}$$

Assuming a high sampling frequency and a slow-varying system, the incremental model as in Eq. (7), provides a linear, time-varying approximation of the nonlinear system [15, 33]. An online system identification algorithm is utilized to generate estimates of the time-varying state and input matrix, $\hat{\boldsymbol{F}}_{t-1} \approx \boldsymbol{F}_{t-1}$ and $\hat{\boldsymbol{G}}_{t-1} \approx \boldsymbol{G}_{t-1}$

*The deadly triad refers to the instability and divergent behavior that methods which combine off-policy learning, bootstrapping, and function approximation, exhibit [22].

3. Network Topology

In this paper single-hidden-layer, fully-connected Multilayer Perceptrons (MLPs) ANNs are chosen as parametric structures for the actor, critic, and target critic, as they: (1) easily manage dimensional large input and output spaces (2) support batch or incremental learning methods (3) are differentiable (4) can approximate any nonlinear function on a compact space arbitrarily well (5) support flexible design (6) are widely applied in intelligent flight control applications [26–28, 30, 34–40]. The hyperbolic tangent activation function is utilized and the hidden layers consist of 10 neurons.

In [26, 27] an additional, offline-trained, trim network is employed to provide a global mapping between the nominal control positions and the system's operating condition. The actor in the framework proposed in this paper is able to learn a notion of a local flight-condition-dependent tracking policy, eliminating the need for a pretrained trim network. Consequently, the input of the actor, critic, and target critic networks include both the state vector and reference state tracking error $\begin{bmatrix} \boldsymbol{s} & \boldsymbol{Ps} - \boldsymbol{s}^R \end{bmatrix}^T \in \mathbb{R}^{(m+p)\times 1}$.

Whereas the input layers of the actor, critic, and target critic have the same structure, their output layers are different. The actor's output layer utilizes a hyperbolic tangent function, which is scaled according to the individual saturation limits of the control surfaces of the PH-LAB research aircraft as presented in Section III. In the case of the elevator for which the saturation limits are asymmetric, the limit with the largest absolute magnitude is utilized as scaling constant. The critic and target critic have a linear output layer. The topology of the neural networks is illustrated in Fig. 3.

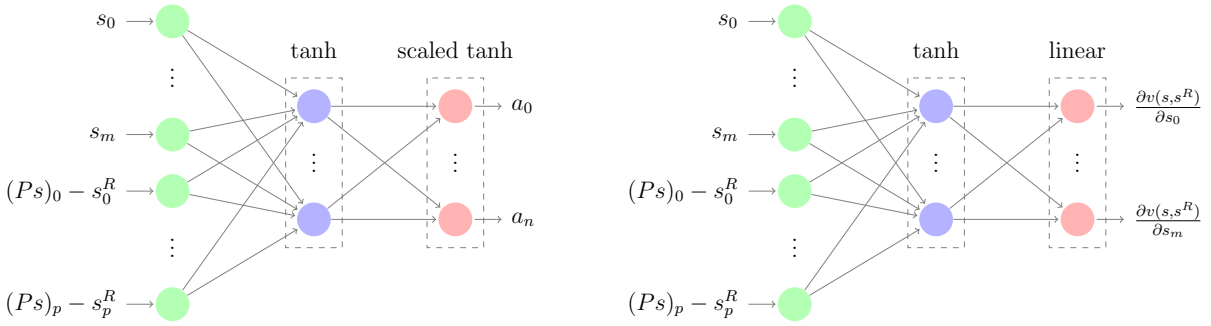

Fig. 3 Neural network topology used for actor (left) and network topology used for critic and target critic (right), with input, hidden, and output layers colored in green, blue, red, respectively. Layers can have either a (scaled) hyperbolic tangent or a linear activation function. The hidden layers have 10 neurons.

C. Update Rules

The parametric structures are updated at each time step as new information becomes available through an interaction with the plant. The leaning process is governed by the update rules, as presented in this section.

1. Critic

The critic is updated through a bootstrapping, TD backup operation. Equation (8) defines the mean squared error loss of the critic L^C to be minimized, with the error $\boldsymbol{e} \in \mathbb{R}^{1\times m}$. The error $\boldsymbol{e}$ is defined as the partial derivative of the TD error with respect to the state vector, as in Eq. (9).

$$L_t^C = \frac{1}{2}\boldsymbol{e}_t\boldsymbol{e}_t^T \tag{8}$$

$$\begin{aligned} \boldsymbol{e}_t &= -\frac{\partial\left[r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1}) + \gamma v(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+1}^R) - v(\boldsymbol{s}_t, \boldsymbol{s}_t^R)\right]}{\partial \boldsymbol{s}_t} \\ &= -\left[\frac{\partial r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1})}{\partial \boldsymbol{s}_{t+1}} + \gamma\hat{\lambda}'(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+1}^R, \boldsymbol{w}_t^{C'})\right]\frac{\partial \boldsymbol{s}_{t+1}}{\partial \boldsymbol{s}_t} + \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{s}_t^R, \boldsymbol{w}_t^C) \end{aligned} \tag{9}$$

In Eq. (9) the target $\left[\frac{\partial r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1})}{\partial \boldsymbol{s}_{t+1}} + \gamma\hat{\lambda}'(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+1}^R, \boldsymbol{w}_t^{C'})\right]\frac{\partial \boldsymbol{s}_{t+1}}{\partial \boldsymbol{s}_t}$ is computed by means of an evaluation of the reward

process, a forward pass through the target critic, a backward pass through the actor, and the current state and input matrix estimates of the incremental model. The latter becomes evident through the expansion of the term $\frac{\partial \boldsymbol{s}_{t+1}}{\partial \boldsymbol{s}_t}$ as presented in Eq. (10). As ANNs are inherently differentiable, the differentiability requirement of the actor in Eq. (10) is met.

$$\begin{aligned}\frac{\partial \boldsymbol{s}_{t+1}}{\partial \boldsymbol{s}_t} &= \frac{\partial f(\boldsymbol{s}_t, \boldsymbol{a}_t)}{\partial \boldsymbol{s}_t} + \frac{\partial f(\boldsymbol{s}_t, \boldsymbol{a}_t)}{\partial \boldsymbol{a}_t}\frac{\partial \boldsymbol{a}_t}{\partial \boldsymbol{s}_t} \\ &= \hat{\boldsymbol{F}}_{t-1} + \hat{\boldsymbol{G}}_{t-1}\frac{\partial \hat{\pi}(\boldsymbol{s}_t, \boldsymbol{s}_t^R, \boldsymbol{w}_t^A)}{\partial \boldsymbol{s}_t}\end{aligned} \tag{10}$$

Equation (11) defines the gradient of the critic's loss with respect to its parameter vector. [†] The critic is updated by stepping in opposite direction to the gradient of the loss with learning rate η^C, as defined in Eq. (12).

$$\frac{\partial L_t^C}{\partial \boldsymbol{w}_t^C} = \frac{\partial L_t^C}{\partial \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{s}_t^R, \boldsymbol{w}_t^C)}\frac{\partial \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{s}_t^R, \boldsymbol{w}_t^C)}{\partial \boldsymbol{w}_t^C} = \boldsymbol{e}_t\frac{\partial \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{s}_t^R, \boldsymbol{w}_t^C)}{\partial \boldsymbol{w}_t^C} \tag{11}$$

$$\boldsymbol{w}_{t+1}^C = \boldsymbol{w}_t^C - \eta_t^C\frac{\partial L_t^C}{\partial \boldsymbol{w}_t^C} \tag{12}$$

2. Target Critic

For the target critic, soft target updates are utilized as first introduced in [32]. The target critic is initialized as a copy of the critic and is subsequently updated according to a weighted sum as defined by Eq. (13). With the scalar mixing factor $\tau \ll 1$. The use of the target critic improves the learning stability as the target state-value derivatives are constrained to change slowly.

$$\boldsymbol{w}_{t+1}^{C'} = \tau\boldsymbol{w}_{t+1}^C + [1-\tau]\boldsymbol{w}_t^{C'} \tag{13}$$

3. Actor

The actor is updated towards an optimal policy $\pi^*(\boldsymbol{s}, \boldsymbol{s}^R) \in \mathbb{R}^{n\times 1}$, as defined in Eq. (14), which maximizes the state-value function $v(\boldsymbol{s}, \boldsymbol{s}^R)$. Consequently, the loss of the actor L^A and its partial derivative with respect to its parameter vector are defined by Eq. (15) and Eq. (16), respectively. Accordingly, the update procedure of the actor involves, the reward process, the critic, and the input matrix estimate of the incremental model. Equation (17) defines the gradient step with learning rate η^A.

$$\pi^*(\boldsymbol{s}_t, \boldsymbol{s}_t^R) = \arg\max_{a_t} v(\boldsymbol{s}_t\boldsymbol{s}_t^R) \tag{14}$$

$$L_t^A = -v(\boldsymbol{s}_t, \boldsymbol{s}_t^R) = -\Big[r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1}) + \gamma v(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+1}^R)\Big] \tag{15}$$

$$\begin{aligned}\frac{\partial L_t^A}{\partial \boldsymbol{w}_t^A} &= -\frac{\partial\Big[r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1}) + \gamma v(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+1}^R)\Big]}{\partial \boldsymbol{w}_t^A} \\ &= -\left[\frac{\partial r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1})}{\partial \boldsymbol{s}_{t+1}} + \gamma\frac{\partial v(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+1}^R)}{\partial \boldsymbol{s}_{t+1}}\right]\frac{\partial \boldsymbol{s}_{t+1}}{\partial \boldsymbol{a}_t}\frac{\partial \boldsymbol{a}_t}{\partial \boldsymbol{w}_t^A} \\ &= -\left[\frac{\partial r(\boldsymbol{s}_t^R, \boldsymbol{s}_{t+1})}{\partial \boldsymbol{s}_{t+1}} + \gamma\hat{\lambda}(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+1}^R, \boldsymbol{w}_{t+1}^C)\right]\hat{\boldsymbol{G}}_{t-1}\frac{\partial \hat{\pi}(\boldsymbol{s}_t, \boldsymbol{s}_t^R, \boldsymbol{w}_t^A)}{\partial \boldsymbol{w}_t^A}\end{aligned} \tag{16}$$

$$\boldsymbol{w}_{t+1}^A = \boldsymbol{w}_t^A - \eta_t^A\frac{\partial L_t^A}{\partial \boldsymbol{w}_t^A} \tag{17}$$

[†] While the target term is also dependent on $\boldsymbol{w}_t^C$ through the update rules of the target critic, this is generally neglected [32, 41]

4. Incremental Model

The estimates of the state and input matrices are represented by the parameter matrix $\hat{\boldsymbol{\Theta}} \in \mathbb{R}^{(m+n)\times m}$, as presented in Eq. (18). The parameter matrix is accompanied by a covariance matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{(m+n)\times(m+n)}$, which expresses a measure of confidence of the estimates. The update process of the Recursive Least Squares (RLS) estimator starts with a prediction of state increments $\Delta\hat{\boldsymbol{s}}$ based on the latest measurements $\boldsymbol{X} \in \mathbb{R}^{(m+n)\times 1}$ and the current parameter estimates $\hat{\boldsymbol{\Theta}}$, as defined in Eq. (19) and Eq. (20). Subsequently, the prediction error $\boldsymbol{\epsilon} \in \mathbb{R}^{1\times m}$, named innovation, is computed, as defined in Eq. (21). Conclusively, the parameter estimates and covariance matrix are updated based on Eq. (22) and Eq. (23), with the scalar forgetting factor $\kappa \in [0, 1]$, which exponentially weights older measurements.

$$\hat{\boldsymbol{\Theta}}_{t-1} = \begin{bmatrix} \hat{\boldsymbol{F}}_{t-1}^T \\ \hat{\boldsymbol{G}}_{t-1}^T \end{bmatrix} \tag{18}$$

$$\boldsymbol{X}_t = \begin{bmatrix} \Delta\boldsymbol{s}_t \\ \Delta\boldsymbol{a}_t \end{bmatrix} \tag{19}$$

$$\Delta\hat{\boldsymbol{s}}_{t+1}^T = \boldsymbol{X}_t^T \hat{\boldsymbol{\Theta}}_{t-1} \tag{20}$$

$$\boldsymbol{\epsilon}_t = \Delta\boldsymbol{s}_{t+1}^T - \Delta\hat{\boldsymbol{s}}_{t+1}^T \tag{21}$$

$$\hat{\boldsymbol{\Theta}}_t = \hat{\boldsymbol{\Theta}}_{t-1} + \frac{\boldsymbol{\Lambda}_{t-1}\boldsymbol{X}_t}{\kappa + \boldsymbol{X}_t^T\boldsymbol{\Lambda}_{t-1}\boldsymbol{X}_t}\boldsymbol{\epsilon}_t \tag{22}$$

$$\boldsymbol{\Lambda}_t = \frac{1}{\kappa}\left[\boldsymbol{\Lambda}_{t-1} - \frac{\boldsymbol{\Lambda}_{t-1}\boldsymbol{X}_t\boldsymbol{X}_t^T\boldsymbol{\Lambda}_{t-1}}{\kappa + \boldsymbol{X}_t^T\boldsymbol{\Lambda}_{t-1}\boldsymbol{X}_t}\right] \tag{23}$$

D. Training Strategy

Algorithm 1 outlines the training strategy of the agent. The proposed strategy targets minimal model-dependency and computational complexity. In [24, 26] the agent's update operations are (partially) conducted on state predictions computed with a model of the plant. Although this approach enables the use of additional optimization schemes at each time step, it has limited applicability to time-critical, online operations, such as the PH-LAB research aircraft. Furthermore, these methods strongly rely on both the model's prediction performance and its capability to estimate the state-transition derivatives.

The method proposed in this paper utilizes current state measurements for the update operations instead. This reduces the model-dependency to only the state-transition derivatives, making the controller's performance less vulnerable to imperfect models. In addition, the computational complexity of the learning algorithm is improved, as it requires less (re)evaluations of the actor, critic, and target critic.

III. Controller Design

This section starts by introducing the simulation model of the PH-LAB research aircraft utilized in this paper. Subsequently, the flight controller design is presented, followed by the hyperparameters.

A. High-Fidelity Simulation Model

Operated by the Faculty of Aerospace Engineering of the Delft University of Technology, the Cessna 550 Citation II PH-LAB depicted in Fig. 1 is a multipurpose research platform. In this paper, a high-fidelity, nonlinear, six-degrees-of-freedom model of a Cessna 500 Citation I, developed with the Delft University Aircraft Simulation Model and Analysis Tool (DASMAT), is utilized. Despite the differences of the aircraft in fuselage size, engine power, and wing size, the model is still representative of the PH-LAB research aircraft [42]. Additionally, the simulation model includes engine dynamics, actuator dynamics, and sensor models. The sensor models are not used, as clean measurements are assumed in this paper. The actuator dynamics are modeled with a first order actuator model and control surface deflection saturation. The limits are listed in Table 1. Furthermore, the aircraft's yaw damper is disabled to provide the agent with full control authority over the control surfaces. The engine's thrust setting is controlled by an internal

Algorithm 1 Learning Framework

Require:

simulation parameters $\Delta t, T$

agent parameters $\gamma, \eta^A, \eta^C, \tau, \kappa$

differentiable deterministic policy parameterization $\hat{\pi}(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^A)$

differentiable state-value function derivative parameterization $\hat{\lambda}(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^C)$

differentiable target state-value function derivative parameterization $\hat{\lambda}'(\boldsymbol{s}, \boldsymbol{s}^R, \boldsymbol{w}^{C'})$

reward function derivative $\frac{\partial r(\boldsymbol{s}^R, \boldsymbol{s})}{\partial \boldsymbol{s}}$

Initialize:

$\boldsymbol{w}_0^A, \boldsymbol{w}_0^C, \hat{\boldsymbol{\Theta}}_0, \boldsymbol{\Lambda}_0, \boldsymbol{s}_0$

$\boldsymbol{w}_0^{C'} \leftarrow \boldsymbol{w}_0^C$

Compute:

for $i = 0$ **to** $int\left(\frac{T}{\Delta t}\right) - 1$ **do**

  get $\boldsymbol{s}_i^R$

  if $i = 1$ **then**

    $\boldsymbol{w}_i^A \leftarrow \boldsymbol{w}_{i-1}^A$

    $\boldsymbol{w}_i^C \leftarrow \boldsymbol{w}_{i-1}^C$

    $\boldsymbol{w}_i^{C'} \leftarrow \boldsymbol{w}_{i-1}^{C'}$

  end if

  if $i > 1$ **then**

    $\begin{bmatrix} \hat{\boldsymbol{F}}_{i-2}^T \\ \hat{\boldsymbol{G}}_{i-2}^T \end{bmatrix} \leftarrow \hat{\boldsymbol{\Theta}}_{i-2}$

    $\frac{\partial \boldsymbol{s}_i}{\partial \boldsymbol{s}_{i-1}} \leftarrow \hat{\boldsymbol{F}}_{i-2} + \hat{\boldsymbol{G}}_{i-2} \frac{\partial \hat{\pi}(\boldsymbol{s}_{i-1}, \boldsymbol{s}_{i-1}^R, \boldsymbol{w}_{i-1}^A)}{\partial \boldsymbol{s}_{i-1}}$

    $\frac{\partial r_i}{\partial \boldsymbol{s}_i} \leftarrow \frac{\partial r(\boldsymbol{s}_{i-1}^R, \boldsymbol{s}_i)}{\partial \boldsymbol{s}_i}$

    $\hat{\boldsymbol{\lambda}}_{i-1} \leftarrow \hat{\lambda}(\boldsymbol{s}_{i-1}, \boldsymbol{s}_{i-1}^R, \boldsymbol{w}_{i-1}^C)$

    $\hat{\boldsymbol{\lambda}}_i \leftarrow \hat{\lambda}(\boldsymbol{s}_i, \boldsymbol{s}_i^R, \boldsymbol{w}_{i-1}^C)$

    $\hat{\boldsymbol{\lambda}}_i' \leftarrow \hat{\lambda}'(\boldsymbol{s}_i, \boldsymbol{s}_i^R, \boldsymbol{w}_{i-1}^{C'})$

    $\Delta \boldsymbol{w}_i^A \leftarrow \eta^A \left[\frac{\partial r_i}{\partial \boldsymbol{s}_i} + \gamma \hat{\boldsymbol{\lambda}}_i\right] \hat{\boldsymbol{G}}_{i-2} \frac{\partial \hat{\pi}(\boldsymbol{s}_{i-1}, \boldsymbol{s}_{i-1}^R, \boldsymbol{w}_{i-1}^A)}{\partial \boldsymbol{w}_{i-1}^A}$

    $\Delta \boldsymbol{w}_i^C \leftarrow -\eta^C \left[-\left[\frac{\partial r_i}{\partial \boldsymbol{s}_i} + \gamma \hat{\boldsymbol{\lambda}}_i'\right] \frac{\partial \boldsymbol{s}_i}{\partial \boldsymbol{s}_{i-1}} + \hat{\boldsymbol{\lambda}}_{i-1}\right] \frac{\partial \hat{\boldsymbol{\lambda}}_{i-1}}{\partial \boldsymbol{w}_{i-1}^C}$

    $\boldsymbol{w}_i^A \leftarrow \boldsymbol{w}_{i-1}^A + \Delta \boldsymbol{w}_i^A$

    $\boldsymbol{w}_i^C \leftarrow \boldsymbol{w}_{i-1}^C + \Delta \boldsymbol{w}_i^C$

    $\boldsymbol{w}_i^{C'} \leftarrow \tau \boldsymbol{w}_i^C + [1 - \tau] \boldsymbol{w}_{i-1}^{C'}$

    $\Delta \boldsymbol{s}_{i-1} \leftarrow \boldsymbol{s}_{i-1} - \boldsymbol{s}_{i-2}$

    $\Delta \boldsymbol{a}_{i-1} \leftarrow \boldsymbol{a}_{i-1} - \boldsymbol{a}_{i-2}$

    $\Delta \boldsymbol{s}_i \leftarrow \boldsymbol{s}_i - \boldsymbol{s}_{i-1}$

    $\boldsymbol{X}_{i-1} \leftarrow \begin{bmatrix} \Delta \boldsymbol{s}_{i-1} \\ \Delta \boldsymbol{a}_{i-1} \end{bmatrix}$

    $\Delta \hat{\boldsymbol{s}}_i^T \leftarrow \boldsymbol{X}_{i-1}^T \hat{\boldsymbol{\Theta}}_{i-2}$

    $\boldsymbol{\epsilon}_{i-1} \leftarrow \Delta \boldsymbol{s}_i^T - \Delta \hat{\boldsymbol{s}}_i^T$

    $\hat{\boldsymbol{\Theta}}_{i-1} \leftarrow \hat{\boldsymbol{\Theta}}_{i-2} + \frac{\boldsymbol{\Lambda}_{i-2} \boldsymbol{X}_{i-1}}{\kappa + \boldsymbol{X}_{i-1}^T \boldsymbol{\Lambda}_{i-2} \boldsymbol{X}_{i-1}} \boldsymbol{\epsilon}_{i-1}$

    $\boldsymbol{\Lambda}_{i-1} \leftarrow \frac{1}{\kappa} \left[\boldsymbol{\Lambda}_{i-2} - \frac{\boldsymbol{\Lambda}_{i-2} \boldsymbol{X}_{i-1} \boldsymbol{X}_{i-1}^T \boldsymbol{\Lambda}_{i-2}}{\kappa + \boldsymbol{X}_{i-1}^T \boldsymbol{\Lambda}_{i-2} \boldsymbol{X}_{i-1}}\right]$

  end if

  $\boldsymbol{a}_i \leftarrow \hat{\pi}(\boldsymbol{s}_i, \boldsymbol{s}_i^R, \boldsymbol{w}_i^A)$

  get $\boldsymbol{s}_{i+1}$ by taking action $\boldsymbol{a}_i$

end for

airspeed controller. The simulation model is run with a sampling frequency of 50 Hz. It contains and accepts a large number of states and control inputs, respectively. A subset of it spans the agent-plant interface as defined by Eq. (24).

Table 1 Aerodynamic control surface saturation limits utilized in the actuator dynamics model of the Cessna 500 Citation I simulation model.

Control Surface	Saturation Limits
δ_e	$[-20.05, 14.90]$ deg
δ_a	$[-37.24, 37.24]$ deg
δ_r	$[-21.77, 21.77]$ deg

$$\boldsymbol{s} = \begin{bmatrix} p & q & r & V_{tas} & \alpha & \beta & \phi & \theta & H \end{bmatrix}^T \quad \boldsymbol{a} = \begin{bmatrix} \delta_e & \delta_a & \delta_r \end{bmatrix}^T \tag{24}$$

B. Flight Controller

The adaptive learning framework is applied to angular rate control of the pitch and roll rate, augmented with an outer control loop, as illustrated in Fig. 4. Rate control exhibits the lowest learning complexity due to the direct dynamic relation between the angular rates and control surfaces. The outer control loop consists of conventional PID controllers and provides a higher-level control interface, which enables reference tracking of an altitude and roll angle profile. Under the assumption of a symmetric aircraft, a decoupled controller design is employed utilizing separate longitudinal and lateral learning controllers, with the state, reference state, and actions vectors as in Eq. (25) and Eq. (26). The resulting selection and weight matrices are presented in Eq. (27) and Eq. (28). For large roll angles, there is limited control of the flight path angle through the pitch rate. On the other hand, using separate controllers with each their own learning framework instance reduces the complexity of the problem to be learned and simplifies the online incremental model identification. As separate critics are used, there is no need to specify relative weights between the longitudinal and lateral tracking errors.

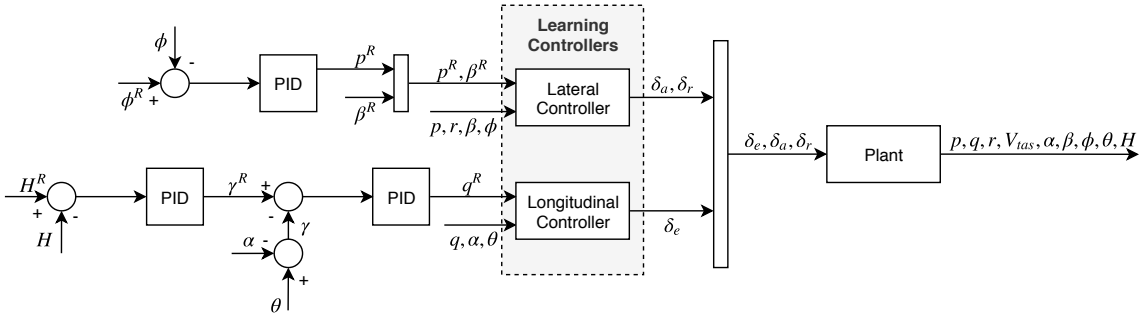

Fig. 4 Schematic of flight controller with individual longitudinal and lateral learning controllers.

$$\boldsymbol{s}^{lon} = \begin{bmatrix} q & \alpha & \theta \end{bmatrix}^T \quad \boldsymbol{s}^{R^{lon}} = \begin{bmatrix} q^R \end{bmatrix}^T \quad \boldsymbol{a}^{lon} = \begin{bmatrix} \delta_e \end{bmatrix}^T \tag{25}$$

$$\boldsymbol{s}^{lat} = \begin{bmatrix} p & r & \beta & \phi \end{bmatrix}^T \quad \boldsymbol{s}^{R^{lat}} = \begin{bmatrix} p^R & \beta^R \end{bmatrix}^T \quad \boldsymbol{a}^{lat} = \begin{bmatrix} \delta_a & \delta r \end{bmatrix}^T \tag{26}$$

$$\boldsymbol{P}^{lon} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad \boldsymbol{Q}^{R^{lon}} = \begin{bmatrix} 1 \end{bmatrix} \tag{27}$$

$$\boldsymbol{P}^{lat} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \boldsymbol{Q}^{R^{lat}} = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix} \tag{28}$$

As can be observed from Eq. (25) the airspeed is not provided to the longitudinal controller. The exclusion of the airspeed is motivated by the fact that the online identification of the airspeed related parameters of the incremental model is nontrivial due to time scale separation and relatively small local variations in airspeed. The latter is exacerbated by the internal airspeed controller. Additional to a reference roll rate, the lateral learning controller receives a zero degree reference for the sideslip angle, such that it can learn to minimize it, with the goal of flying coordinated turns.

C. Hyperparameters

The hyperparameters utilized in the remainder of this paper are summarized in Table 2. These parameters are utilized for both the longitudinal and lateral controllers and are determined empirically by experimentation and common guidelines proposed in [24]. In Section IV a distinction is made between an online training, online operation, and online adaption phase. As the name suggests, the training phase focuses on the learning process of the controllers. During the operation phase, the agent is employed to fly representative maneuvers. In the adaption phase, changes are applied to the plant and the agent during flight. In all phases, the agent is constantly learning through interaction. During the training phase, a faster learning process is preferred over a lower chance of converging to a local minimum. Accordingly, the learning rates during the training phase are higher, than in the operation phase. The utilization of the target critic depends on the experiment being conducted in Section IV. For the training phase the RLS incremental model estimator, the actor, and the critic are initialized according to the values as listed in Table 2. During the operation phase, the learning framework is initialized at its pretrained state.

The forgetting factor κ of the RLS estimator is set to one. This is done to ensure consistency of the estimator. During periods of poor or no excitation (which is the case for the majority of time during operation), the covariance matrix increases exponentially when $\kappa < 1$, also referred to as estimator windup. Once, the system is excited again, abrupt changes in the estimate occur despite no changes in the actual system. In addition to poor excitation, non-uniformly distributed information over all parameters and time-scale separation in the variation of parameters also lead to estimator windup. Many approaches to mitigate these issues have been proposed. In [43–45] non-uniformly distributed information is dealt with by selective amplification of the covariance matrix, called directional forgetting. Similarly, [46, 47] propose vector-type or selective forgetting, were individual parameter dependent forgetting factors are used, to deal with the effects of time-scale separation. The design of an advanced online parameter identification algorithm for the PH-LAB is out of the scope of this research.

Setting the forgetting factor to one has the disadvantage that the estimator does not forget older data. Consequently, the estimator becomes less adaptive over time. In this paper, this problem is mitigated by resetting the covariance matrix to its initial value when a large change in the system is detected through the estimator's innovation term, as elaborated in Section IV.

Table 2 Hyperparameters of learning framework instances of both lateral and longitudinal controllers.

Parameter	Value
η^A, η^C, τ	training phase $5, 10, 0.01$ or $5, 10, 1$ operation phase $1, 2, 0.01$
$\boldsymbol{w}_0^A, \boldsymbol{w}_0^C$	$\mathcal{N}_{trunc}(\mu = 0, \sigma = 0.05)$
$\hat{\boldsymbol{F}}_0, \hat{\boldsymbol{G}}_0, \Lambda_0$	$I, \mathbf{0}, I \cdot 10^8$
κ	1.0
γ	0.8

IV. Results and Discussion

In this section, the results of the conducted experiments are presented and discussed. Three distinct cases are evaluated. First, the framework's capability to train online without a priori knowledge of the system dynamics is evaluated. In addition, the online identification of the incremental model, as well as, the effect of the target critic on the learning stability are discussed. Subsequently, experiments are conducted to demonstrate that the agent is able to operate the aircraft during representative maneuvers at a variety of flight conditions. Last but not least, the framework's capability of dealing with unforeseen changes, such as changes in the aircraft's dynamics, is analyzed.

A. Online Training Phase

In this section, the framework's capability of online learning without a priori knowledge of the plant is demonstrated. The longitudinal and lateral controllers are trained separately. During training of the longitudinal controller, the control surfaces associated with lateral motion are kept at their initial trim value and vice versa.

A sine function with an amplitude of 5 degrees per second and a frequency of 0.2 Hertz is utilized as pitch and roll rate reference. As elaborated in the previous section, the sideslip angle reference is zero. Different signals commonly used for (aircraft) system identification [48], such as frequency sweeps, doublets, 3211 doublets, and sinusoidal functions are good reference signal candidates. The proposed sinusoidal reference allows for a short training phase (under 60 seconds) with an acceptable load factor range, as depicted in Fig. 5. Persistent Excitation (PE) is essential to both state-space exploration in the learning process and dynamic excitation in the system identification process of the incremental model [22, 24, 48]. Exponentially decaying, sinusoidal excitation is applied to the elevator and ailerons to excite the system during the initial training phase. As the agent learns to track the dynamic reference signals, the excitation on the elevator and ailerons is reduced. Otherwise, the agent would learn to compensate for the applied excitation over time. Through the aircraft's dutch roll eigenmode, its yaw motion is also excited through the ailerons. Therefore no additional excitation is applied to the rudder.

As illustrated in Fig. 5 both the longitudinal and lateral controller is able to follow the reference signals after less than 30 seconds of training. The same observation is made from the actor and critic parameters as depicted in Fig. 6, as an overall convergence of the parameters can be observed. As a result of the motion in the longitudinal controller training phase, the airspeed oscillates. The agent perceives changes in airspeed as changes in the environment and constantly adapts to it. This can be observed in Fig. 6, where the actor and critic parameters of the longitudinal controller show minor oscillatory behavior.

From Fig. 5 it can be observed that it takes longer to learn to control the sideslip angle than the angular rates. This can be attributed to two main factors: (1) the control surfaces are more directly related to the angular rates than to the sideslip angle (2) the rudder is not directly excited. The consequence of the latter can also be observed in the online identification process of the incremental models.

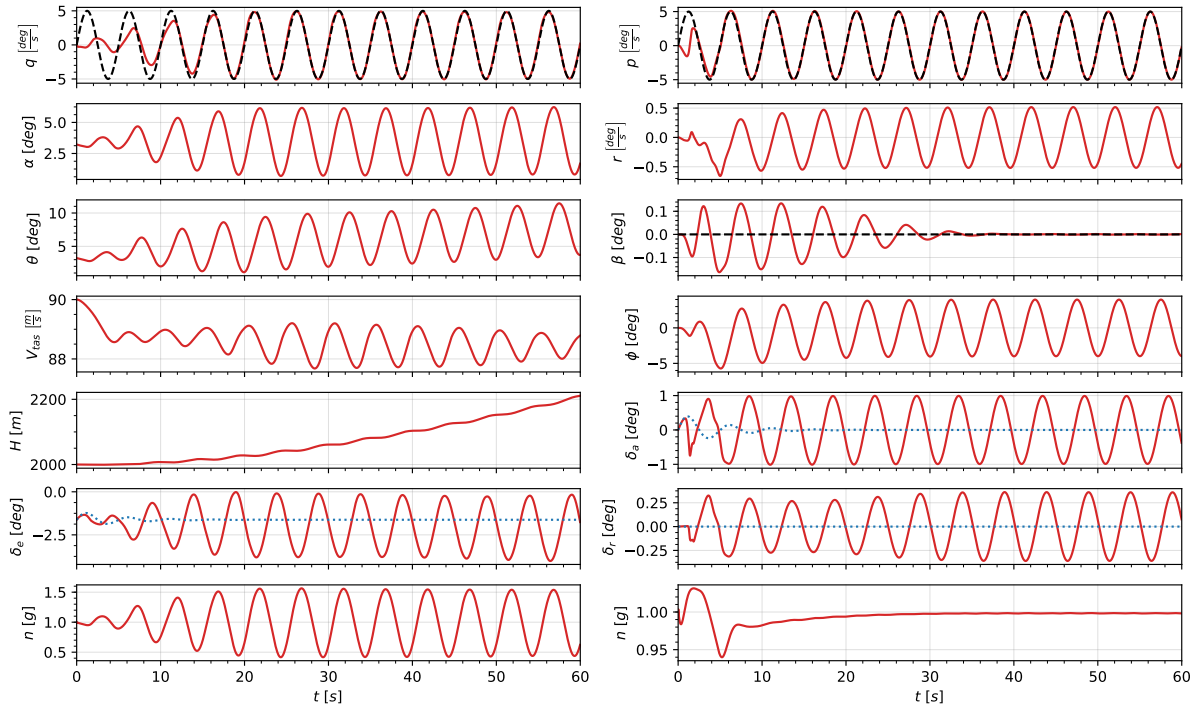

Fig. 5 Online training procedures of longitudinal (left) and lateral (right) controller, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$. Reference and excitation signals are illustrated by black dashed and blue dotted lines, respectively.

1. Online Identification of Incremental Model

Figure 6 depicts the online identified state and input matrices for both the longitudinal and lateral incremental models. As elaborated in Section II, the state and input matrices play an important role in the update operations of both the actor and critic. For meaningful updates of the actor and critic, the online identified model parameters estimates should have the correct sign and to some extent correct relative magnitude. To allow for an evaluation of the identification performance, reference parameters values are provided by dotted lines. These reference values are derived from a linearization of the nonlinear plant at the initial trim condition at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. These constant values are not exact as: (1) they do not take in to account the actuator dynamics (2) nor the airspeed controller (3) do not take into account time-varying changes in the plant (4) the linearized model contains different states than the incremental model's regression matrix. Nonetheless, these reference values provide a good approximation to evaluate the performance of the online identified estimates.

It can be observed that both the parameters of the state and input matrices are identified in less than 10 seconds, providing the agent with local information about the plant, with the exception of one term. The term $\frac{\partial p}{\partial \delta_r}$ of the input matrix of the lateral incremental model, is initially incorrectly identified and slowly moves towards its correct final value. This behavior is partially a consequence of the fact that the rudder is not initially directly excited.

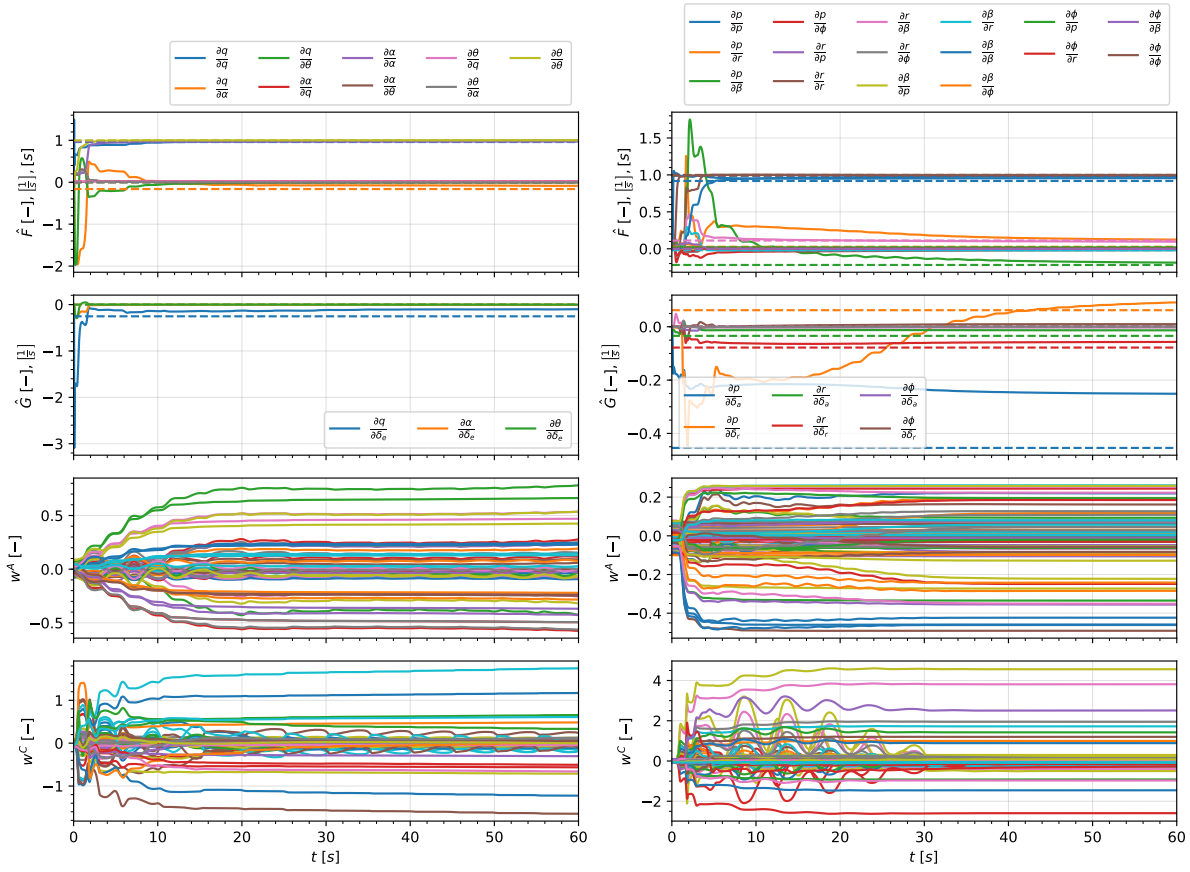

Fig. 6 State and input matrix estimates, and actor and critic parameters during online training of longitudinal (left) and lateral (right) controller, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$ and $(\hat{\boldsymbol{F}}_0, \hat{\boldsymbol{G}}_0, \hat{\boldsymbol{\Theta}}_0) = (I, \mathbf{0}, I \cdot 10^8)$. Estimates derived from a plant linearization are represented by dotted lines.

2. Online Training with Untrimmed Initialization

The online training phase as presented in the previous section is initiated at a trimmed condition. In practice, however, it is challenging to find an initial trimmed state without a priori knowledge of the plant. Therefore, the proposed framework should also be able to deal with untrimmed initial conditions. The effect of an untrimmed initialization on the training phase is therefore examined for the longitudinal controller by superimposing a random uniformly distributed elevator deflection offset around its trimmed state, in addition to the small random offset originating from the random initialization of the actor's parameters. Furthermore, the effects of the proposed utilization of a target critic on the learning speed and stability are examined, by comparing the results attained from experiments with and without a target critic. 500 runs are simulated for each case.

In Fig. 7 the area between the 25th and 75th percentile of the 500 runs are presented in red and blue, for a mixing factor of $\tau = 1.0$ and $\tau = 0.01$, respectively. In both cases the same learning rates are utilized $(\eta^A, \eta^C) = (5, 10)$. It can be observed that the interquartile range of the agent without target critic narrows within the first 22 seconds of training. In comparison, for the agent with a target critic, this is observed 20 seconds later. This implies that it takes longer for the majority of the runs to converge to an equivalent final policy. On the other hand, in the first 5 seconds of training the interquartile range of the agent with a target critic spans a smaller range of experienced pitch rate values than its counterpart. This implies that for the majority of the 500 runs, the initial training phase is less dynamic and erroneous, assisting the training stability. The latter is confirmed by the failure rates. The sample mean failure rate for 500 runs for the agent with and without target critic ($\tau = 0.01$ and $\tau = 1.0$) is 0% and 5.2%, respectively. Failure is defined as the occurrence of a float overflow in the learning process due to numerical instability.

Most of the time an unstable or diverging learning behavior leads to fast growth of the critic's parameters. As a consequence, the actor's loss gradients become very large leading to aggressive and large updates from which the agent cannot recover. The utilization of the proposed target critic stabilizes the critic's learning and as a consequence, no failures are encountered within 500 runs. As a comparison, without the proposed target critic, the training of the longitudinal controller has a failure rate of 5.2%.

Consistent with findings in [32] the target critic slows down and stabilizes the learning process. Especially in an application in the PH-LAB aircraft, no failure is acceptable. Therefore, the augmentation of the current IDHP framework with the proposed target critic is vital to a successful implementation of the RL controller to the PH-LAB.

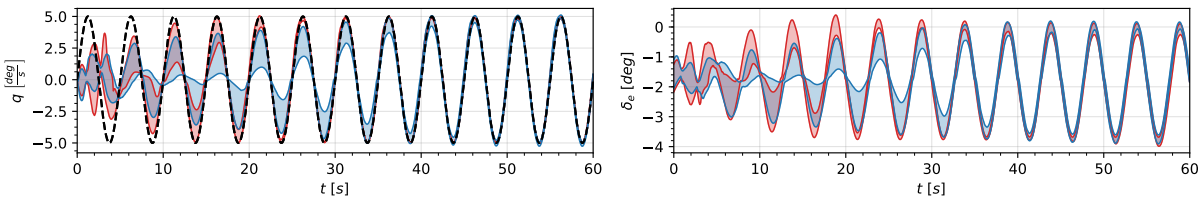

Fig. 7 Interquartile range of pitch rate and elevator deflection for 500 runs of the online training phase of the longitudinal controller with random untrimmed initialization around the operating point $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. The reference signal is represented by a black, dashed line. The agent with and without a target critic are represented in blue and red, respectively.

B. Online Operation Phase

Subsequent to a successful online training phase of both the lateral and longitudinal controllers, these can be utilized to operate the aircraft. In this experiment, the goal is to demonstrate that the agent is able to operate the aircraft, successfully and reliably during representative maneuvers at a variety of flight conditions. For each of the four initial flight conditions as listed in Table 3, the agent follows an altitude and bank angle flight profile, which consists of a prolonged climb and a subsequent descent (with 5 $\frac{m}{s}$), combined with left and right turns (of 25 deg and 20 deg), as depicted in Fig. 8 and Fig. 9. For each flight condition 100 runs are simulated, where the plant is initialized at a trimmed state and the agent in its final state of the online training phase of the previous section.

Figure 8 illustrates all 100 flight maneuvers for flight condition FC0, starting at the same condition in which the previous online training phase was conducted. For all states and actions, as well as, the reference pitch and roll rate, the figure depicts the area between the minimum and maximum bounds of all 100 runs. The bounds emphasize that none of the 100 runs experience a failure, as listed in Table 3. It can be observed that the agent flies the commanded height and

bank angle profile, by following the reference pitch and roll rate, as commanded by the PID controllers. In addition, the agent succeeds to minimize the sideslip angle and conduct a well-coordinated turn, by actuating the rudder.

Although the online pretrained agent already succeeds in flying the aircraft, its performance continues to improve. From the min-max bound of the time history of the sideslip angle and rudder deflection, a minor oscillatory behavior is observed in the first two turns. As the agent carries out the first two turns it gains more experience and improves its performance. Consequently, the oscillatory behavior vanishes and min-max bound narrows for the two subsequent turns.

From the time history of the pitch rate and airspeed, it can be observed that the agent's pitch rate reference tracking performance temporarily degrades during quick changes in airspeed. Due to the separation between the airspeed controller and the longitudinal and lateral controllers, as elaborated in Section III, the learning controllers do not conceive a notion of the airspeed as a distinct state, but can only experience it as an external, temporary change in the perceived plant dynamics. Although the agent's performance temporary degrades, due to quick changes in airspeed, the agent still adapts to different airspeed regimes and their influence on the plant's dynamics.

This is demonstrated in Fig. 9, where the agent (pretrained at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$) is utilized to operate at the flight condition FC3 with $(V_{tas}, H) = (140\ \frac{m}{s}, 5\ km)$. Despite the different flight condition, which the agent has not previously experienced, the agent is able to follow the flight profile in all 100 runs without failure, as depicted in by Fig. 9 and Table 3. Similarly, to the previous flight condition, the agent is still constantly adapting and improving upon new experiences as can be observed for example from the decrease in width and magnitude of the min-max bound of the time history of the sideslip angle.

Table 3 Description and sample failure rate of different flight conditions each simulated 100 times during the online operation phase. For all cases the agent is initialized at its final state of the online training phase conducted at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$.

Flight Condition ID	$H_0\ m$	$V_{tas_0}\ \frac{m}{s}$	Failure Rate
FC0	2000	90	0%
FC1	2000	140	0%
FC2	5000	90	0%
FC3	5000	140	0%

C. Online Adaption

During operation, the controller needs to be able to adapt to uncertainties in the system, such as unexpected failures or time-varying components [8, 13, 49, 50]. Conventional ACDs that require an initial offline training phase, are unable to quickly adapt online to these changes [30]. In this section, the adaptability of the proposed framework is validated in two experiments. In the first step, a failure is introduced to the aileron to demonstrate the framework's capability to identify changes in the plant. Subsequently, another experiment is conducted where a disturbance is introduced directly in the actor's parameters. In both cases, the agent is initiated at its online pretrained state. The controller is commanded to fly two right rate-one turns while holding the initial altitude. The disturbances are induced at the start of the second turn at 110 seconds.

1. Adaptive Control in the Presence of Aileron Failure

To simulate the failure of one aileron, the aileron deflection as commanded by the actor is halved before it is passed as input to the plant. As a result, the perceived control effectiveness of the ailerons is halved and the controller has to double its aileron command to fly the turn. In this experiment, the covariance matrix of the lateral incremental model is reinitialized at $I \cdot 10^8$ once a fault is detected. Many methods for fault detection have been proposed in [3, 13, 49]. Here, a sudden increase of the innovation term, as presented in Eq. (21), is utilized to monitor for changes in the plant. As illustrated in Fig. 10 the controller is able to detect the change in the plant and complete the turn, despite the failure in the ailerons. From the estimates of the input matrix of the lateral incremental model, it can be observed that the online identification perceives the change in the plant's dynamics. For example the incremental parameter that corresponds to the $\frac{\partial p}{\partial \delta_a}$ term, changes from the initial value of $0.24\ \frac{1}{s}$ to $0.12\ \frac{1}{s}$.

Consequently, the agent commands a larger aileron deflection, as illustrated in Fig. 10, to complete the right turn. In this case, the control adaption of the agent interacts with the outer loop PID controller. Therefore, in the next experiment,

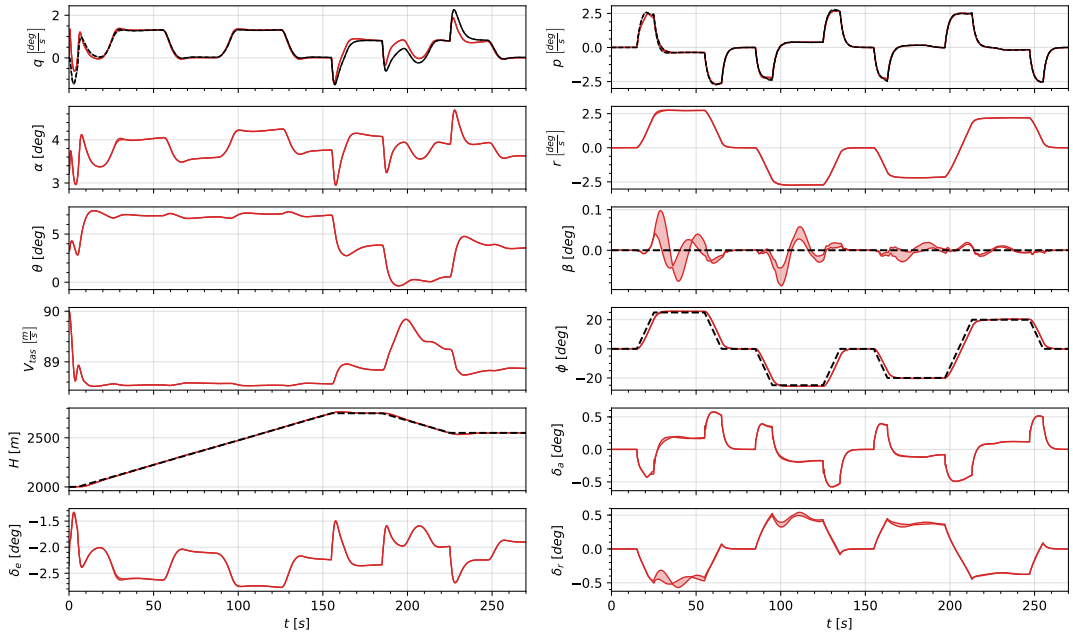

Fig. 8 Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC0 $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines.

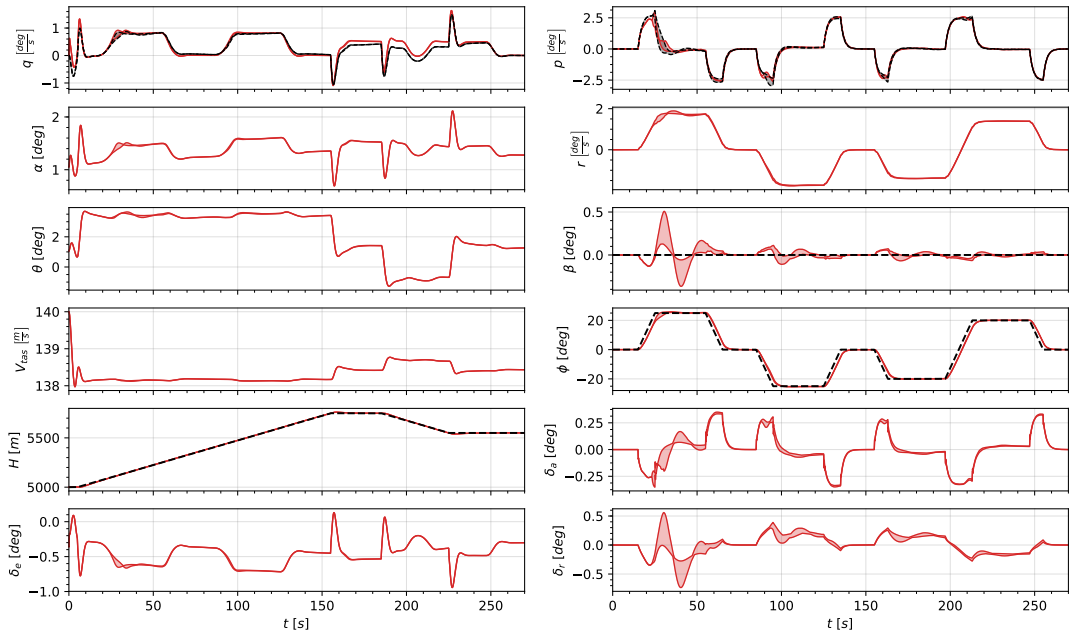

Fig. 9 Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC3 $(V_{tas}, H) = (140\ \frac{m}{s}, 5\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines.

the adaptability of the agent is demonstrated by injecting a disturbance directly on the parameters of the actor.

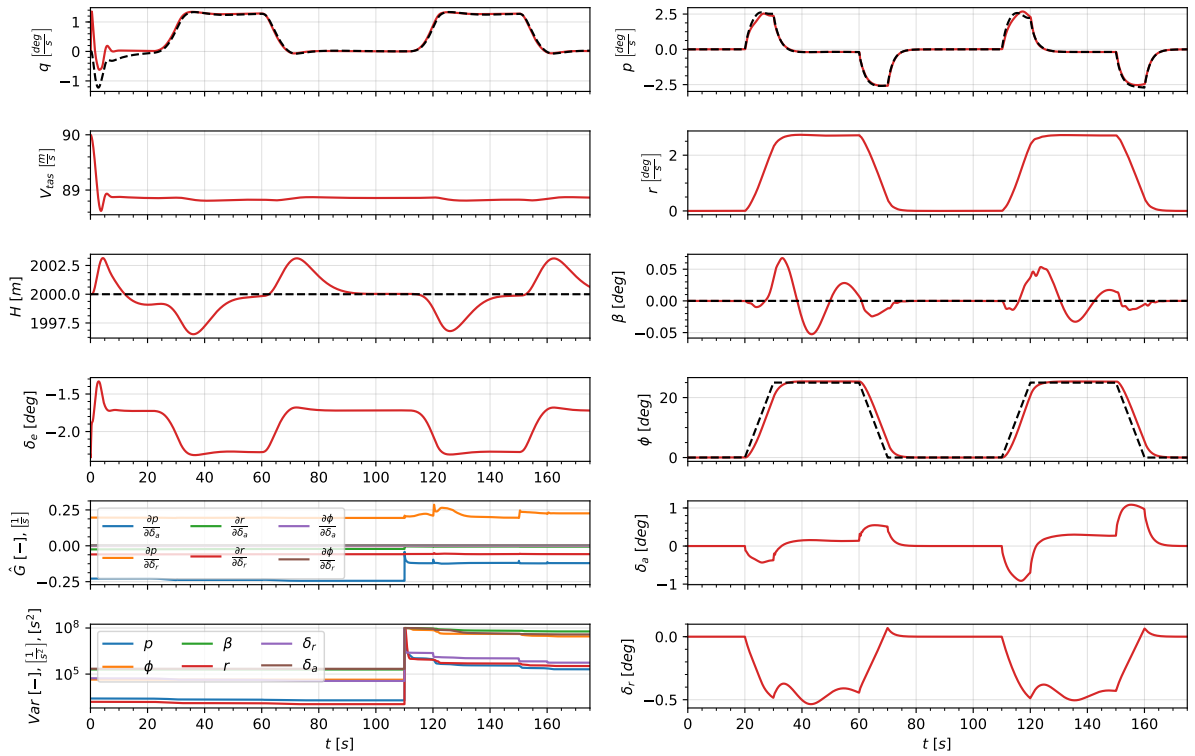

Fig. 10 Control adaption and online identification of the lateral incremental model during an aileron failure induced at 110 seconds. The plant is initiated at a trimmed state at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and the longitudinal and lateral controller at their state after the initial online training phase. Reference signals are represented by black, dashed lines.

2. Adaptive Control in the Presence of Control Disturbance

Similarly to the previous experiment, the agent is commanded to fly two right rate-one turns, while maintaining its initial altitude. At 110 seconds the actor's parameters are disturbed by a zero-mean Gaussian noise with a standard deviation of 0.2. Consequently, the current policy is disturbed and the agent has to adapt to regain control over the aircraft and complete the flight maneuver.

As can be observed from Fig. 11, subsequent to the disturbance at 110 seconds the actor's parameters adapt towards a new near-optimal policy within 10 seconds. This adaption is also observed from the peak in the actor's loss gradients as depicted in the lower left box of Fig. 11. Furthermore, the actor's parameters converge to a different distribution, than its initial one. This implies that there are more than one near-optimal policies and that a near-optimal policy is not uniquely defined by one set of the actor's parameters.

From the time history of the sideslip angle and roll rate, it can be observed that the disturbance is detrimental to the agent's performance. More specifically, both the overall magnitude of the sideslip angle as well as roll rate tracking error increases during the initiation of the second right turn. Nonetheless, the agent still manages to follow the reference roll angle profile. As the agent exits the second turn at 150 seconds it has already recovered and improved upon its original performance.

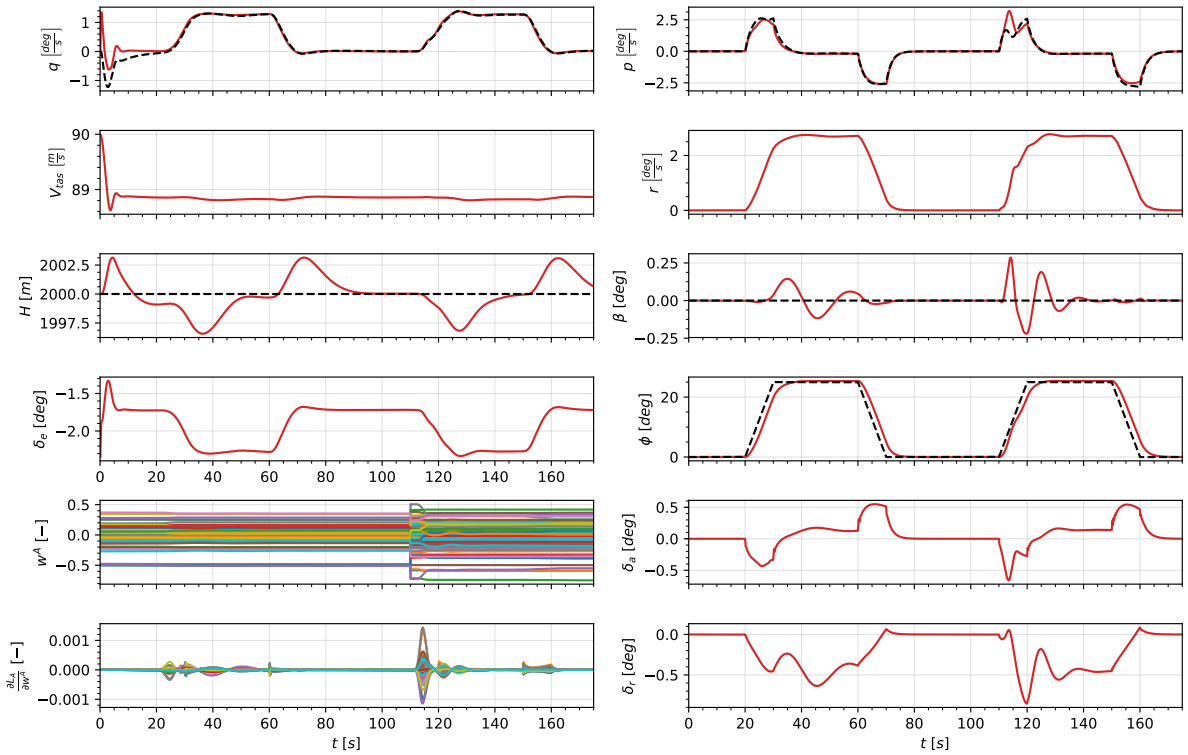

Fig. 11 Control adaption of the lateral controller during a Gaussian disturbance to the actor's parameters at 110 seconds. The plant is initiated at a trimmed state at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and the longitudinal and lateral controller at their state after the initial online training phase. Reference signals are represented by black, dashed lines.

V. Conclusion

The design and analysis of a RL adaptive flight controller for a CS-25 class aircraft are presented. The adaptive, learning controller is successfully implemented for a full-scale, high-fidelity aircraft simulation. It is demonstrated that the proposed framework is able to learn a near-optimal control policy online without a priori knowledge of the system dynamics nor an offline training phase. The results reveal that the proposed target critic increases the learning stability, which is vital for the reliable operation of the aircraft. Through the simulation of representative flight profiles, the results indicate that the learning controller is able to generalize and operate the aircraft in not previously encountered flight regimes as well as identify and adapt to unforeseen changes to the aircraft's dynamics. The framework proposed in this study mitigates the current limitation of ACDs that require an offline training phase expanding the application of such algorithms to applications for which no accurate system model is available, nor readily identifiable. This is especially interesting for future autonomous applications. Further investigation is recommended into the design of information-based online system identification and the execution of a flight test with the proposed framework on the PH-LAB research aircraft.

References

[1] Woods, D. D., "The risks of autonomy: Doyle's catch," *Journal of Cognitive Engineering and Decision Making*, Vol. 10, No. 2, 2016, pp. 131–133.

[2] Sghairi, M., Bonneval, A., Crouzet, Y., Aubert, J. J., and Brot, P., "Challenges in Building Fault-Tolerant Flight Control System for a Civil Aircraft," *IAENG International Journal of Computer Science*, Vol. 35, No. 4, 2008.

[3] Lombaerts, T., Oort, E. V., Chu, Q. P., Mulder, J. A., and Joosten, D., "Online Aerodynamic Model Structure Selection and Parameter Estimation for Fault Tolerant Control," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 3, 2010, pp. 707–723.

[4] Balas, G. J., "Flight Control Law Design: An Industry Perspective," *European Journal of Control*, Vol. 9, No. 2-3, 2003, pp. 207–226.

[5] Lane, S. H., and Stengel, R. F., "Flight control design using non-linear inverse dynamics," *Automatica*, Vol. 24, No. 4, 1988, pp. 471–483.

[6] da Costa, R. R., Chu, Q. P., and Mulder, J. A., "Reentry Flight Controller Design Using Nonlinear Dynamic Inversion," *Journal of Spacecraft and Rockets*, Vol. 40, No. 1, 2003, pp. 64–71.

[7] Sonneveldt, L., van Oort, E. R., Chu, Q. P., De Visser, C. C., Mulder, J. A., and Breeman, J. H., "Lyapunov-based Fault Tolerant Flight Control Designs for a Modern Fighter Aircraft Model," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Chicago, Illinois, 2009.

[8] Farrell, J., Sharma, M., and Polycarpou, M., "Backstepping-Based Flight Control with Adaptive Function Approximation," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 6, 2005, pp. 1089–1102.

[9] Sonneveldt, L., Chu, Q. P., and Mulder, J. A., "Nonlinear Flight Control Design Using Constrained Adaptive Backstepping," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 322–336.

[10] Sonneveldt, L., van Oort, E. R., Chu, Q. P., and Mulder, J. A., "Comparison of Inverse Optimal and Tuning Functions Designs for Adaptive Missile Control," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 4, 2008, pp. 1176–1182.

[11] Sonneveldt, L., van Oort, E. R., Chu, Q. P., and Mulder, J. A., "Nonlinear Adaptive Trajectory Control Applied to an F-16 Model," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 25–39.

[12] van Oort, E. R., Sonneveldt, L., Chu, Q. P., and Mulder, J. A., "Full-Envelope Modular Adaptive Control of a Fighter Aircraft Using Orthogonal Least Squares," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1461–1472.

[13] Lu, P., van Kampen, E., de Visser, C., and Chu, Q. P., "Aircraft fault-tolerant trajectory control using Incremental Nonlinear Dynamic Inversion," *Control Engineering Practice*, Vol. 57, 2016, pp. 126–141.

[14] Sieberling, S., Chu, Q. P., and Mulder, J. A., "Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 6, 2010, pp. 1732–1742.

[15] Simplício, P., Pavel, M. D., van Kampen, E., and Chu, Q. P., "An acceleration measurements-based approach for helicopter nonlinear flight control using incremental Nonlinear Dynamic Inversion," *Control Engineering Practice*, Vol. 21, No. 8, 2013, pp. 1065–1077.

[16] Acquatella, P., Falkena, W., Van Kampen, E., and Chu, Q. P., "Robust Nonlinear Spacecraft Attitude Control using Incremental Nonlinear Dynamic Inversion," *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, Minnesota, 2012.

[17] Smeur, E. J. J., Chu, Q. P., and de Croon, G. C. H. E., "Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 3, 2016, pp. 450–461.

[18] Acquatella, P., van Kampen, E., and Chu, Q. P., "Incremental Backstepping for Robust Nonlinear Flight Control," *Proceedings of the EuroGNC*, Delft, The Netherlands, 2013.

[19] Wang, X., van Kampen, E., Chu, Q. P., and Lu, P., "Stability Analysis for Incremental Nonlinear Dynamic Inversion Control," *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida, 2018.

[20] Grondman, F., Looye, G., Kuchar, R. O., Chu, Q. P., and van Kampen, E., "Design and Flight Testing of Incremental Nonlinear Dynamic Inversion-based Control Laws for a Passenger Aircraft," *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida, 2018.

[21] Schultz, W., Dayan, P., and Montague, P. R., "A Neural Substrate of Prediction and Reward," *Science*, Vol. 275, No. 5306, 1997, pp. 1593–1599.

[22] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction*, 2nd ed., A Bradford Book, 2018.

[23] Powell, W. B., *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, John Wiley & Sons, 2007.

[24] Si, J., Barto, A. G., Powell, W. B., and Wunsch, D., *Handbook of Learning and Approximate Dynamic Programming*, Wiley-IEEE Press, 2004.

[25] Bertsekas, D. P., Homer, M. L., Logan, D. A., Patek, S. D., and Sandell, N. R., "Missile defense and interceptor allocation by neuro-dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol. 30, No. 1, 2000, pp. 42–51.

[26] 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.

[27] Enns, R., and Si, J., "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Transactions on Neural Networks*, Vol. 14, No. 4, 2003, pp. 929–939.

[28] van Kampen, E., Chu, Q. P., and Mulder, J. A., "Continuous Adaptive Critic Flight Control Aided with Approximated Plant Dynamics," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2006.

[29] Zhou, Y., van Kampen, E., and Chu, Q. P., "Launch Vehicle Adaptive Flight Control with Incremental Model Based Heuristic Dynamic Programming," *68th International Astronautical Congress (IAC)*, Adelaide, Australia, 2017.

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

[31] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[32] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous Control with Deep Reinforcement Learning," *International Conference on Learning Representations (ICLR)*, 2016.

[33] Zhou, Y., van Kampen, E., and Chu, Q. P., "Nonlinear Adaptive Flight Control Using Incremental Approximate Dynamic Programming and Output Feedback," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 493–496.

[34] Sadhukhan, D., and Feteih, S., "F8 neurocontroller based on dynamic inversion," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 1, 1996, pp. 150–156.

[35] Napolitano, M. R., and Kincheloe, M., "On-line learning neural-network controllers for autopilot systems," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 5, 1995, pp. 1008–1015.

[36] Kim, B. S., and Calise, A. J., "Nonlinear Flight Control Using Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 26–33.

[37] Calise, A. J., "Neural networks in nonlinear aircraft flight control," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 11, No. 7, 1996, pp. 5–10.

[38] Ha, C. M., "Neural networks approach to AIAA aircraft control design challenge," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 4, 1995, pp. 731–739.

[39] Balakrishnan, S. N., and Biega, V., "Adaptive-Critic-Based Neural Networks for Aircraft Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 4, 1996, pp. 893–898.

[40] Prokhorov, D. V., and Wunsch, D. C., "Adaptive critic designs," *IEEE Transactions on Neural Networks*, Vol. 8, No. 5, 1997, pp. 997–1007.

[41] Srouji, M., Zhang, J., and Salakhutdinov, R., "Structured Control Nets for Deep Reinforcement Learning," *arXiv preprint arXiv:1802.08311*, 2018.

[42] van den Hoek, M. A., de Visser, C. C., and Pool, D. M., "Identification of a Cessna Citation II Model Based on Flight Test Data," *Advances in Aerospace Guidance, Navigation and Control*, Springer International Publishing, Cham, 2018, pp. 259–277.

[43] Hägglund, T., "Recursive Estimation of Slowly Time-Varying Parameters," *IFAC Proceedings Volumes*, Vol. 18, No. 5, 1985, pp. 1137–1142.

[44] Kulhavý, R., "Restricted exponential forgetting in real-time identification," *Automatica*, Vol. 23, No. 5, 1987, pp. 589–600.

[45] Cao, L., and Schwartz, H. M., "A novel recursive algorithm for directional forgetting," *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, Vol. 2, IEEE, 1999, pp. 1334–1338.

[46] Saelid, S., and Foss, B., "Adaptive controllers with a vector variable forgetting factor," *The 22nd IEEE Conference on Decision and Control*, IEEE, 1983, pp. 1488–1494.

[47] Parkum, J. E., Poulsen, N. K., and Holst, J., "Selective Forgetting in Adaptive Procedures," *IFAC Proceedings Volumes*, Vol. 23, No. 8, 1990, pp. 137–142.

[48] Klein, V., and Morelli, E. A., *Aircraft System Identification: Theory and Practice*, American Institute of Aeronautics and Astronautics, 2006.

[49] Lu, P., Van Eykeren, L., van Kampen, E., de Visser, C., and Chu, Q., "Double-model adaptive fault detection and diagnosis applied to real flight data," *Control Engineering Practice*, Vol. 36, 2015, pp. 39–57.

[50] Looye, G., and Joos, H., "Design of robust dynamic inversion control laws using multi-objective optimization," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2001, p. 4285.

II

Literature Survey and Preliminary Analysis

2

Literature Survey

The literature survey as presented in this chapter starts with an introduction to the fundamentals of Reinforcement Learning (RL) in Section 2.1 followed by an overview of the RL approaches for continuous spaces in Section 2.2. Next, research of the RL branches most applicable to the PH-LAB research aircraft are reviewed in Section 2.3 and Section 2.4. Last but not least, the literature survey is concluded in Section 2.5 with the proposal of a baseline RL framework.

2.1. Reinforcement Learning Fundamentals

This section provides a comprehensive introduction to RL, by first presenting its key concepts and terminology, followed by the fundamental ideas utilized to solve them, in Section 2.1.1 and Section 2.1.2, respectively. Subsequently, a set of dimensions that span the framework of RL are presented in Section 2.1.3. By the end of this section the fundamental knowledge basis, required for the discussions in subsequent sections, is established.

2.1.1. Key Concepts

The idea of learning from interaction, the essence of RL, is defined by a framework with specific entities, terminology, and dimensions. The main entity is the *agent* that interacts with the second entity, the *environment*, as depicted in Figure 2.1. The interface through which the these entities interact can be specified by *states*, *actions* and *rewards*. A decision made by the agent is called action and is based on the observation of the environment's state. The environment also provides the agent with feedback through a reward. This feedback forms the basis through which the agent evaluates its decisions. The agent's objective is to maximize the rewards it receives over time. The terms agent, environment and action are also commonly referred to as the controller, plant, and control signal in engineering terms.

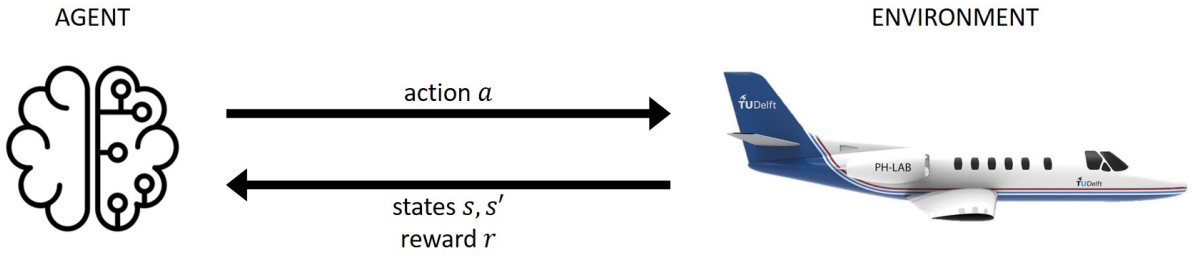

Figure 2.1: Agent-environment interaction in the fundamental framework of RL.

Markov Decision Processes (MDPs)

A mathematically idealized framework to describe this sequential decision making problem is formed with MDPs, where the agent and environment interact in a sequence of discrete time steps $t = 0, 1, 2, \ldots$.

Assuming full observability [1], the agent observes the environment's state $S_t \in \mathcal{S}$, at each time step. Based on that observation the agent selects an action $A_t \in \mathcal{A}(s)$. Partially as consequence of its action, the agent receives a numerical reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and observes the environment's next state $S_{t+1} \in \mathcal{S}$. S_t, A_t and R_{t+1} are random variables. This sequential process gives rise to the trajectory $S_0, A_0, R_1, S_1, A_1, R_2, \ldots$. This framework is further simplified by assuming the sets $\mathcal{S}, \mathcal{A}(s)$ and $\mathcal{R}$ to be finite, hence to be *finite* MDPs. As consequence, S_t and R_t can be described by a discrete probability distribution, for all $s', s \in \mathcal{S}, r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$, as in Equation (2.1). MDPs also assume the Markov property, which as defined in Equation (2.2), states that the current state carries information about all past agent-environment interactions. In other words. The future is independent of the past given the present.

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{2.1}$$

$$\Pr\{S_{t+1} = s'|S_0 = s_0, S_1 = s_1, ..., S_t = s_t\} = \Pr\{S_{t+1} = s'|S_t = s_t\} \tag{2.2}$$

Environment Processes

Equation (2.1) in the framework of MDPs fully describes the environment. Commonly two functions are derived from the environment's processes of state-transition and feedback. The state-transition function, as in Equation (2.3) and the reward function, expressed either by Equation (2.4) or Equation (2.5).

$$p(s'|s, a) \doteq \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a) \tag{2.3}$$

$$r(s, a) \doteq \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a) \tag{2.4}$$

$$r(s, a, s') \doteq \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)} \tag{2.5}$$

Return

As previously stated the goal of the agent is to maximize the reward it receives over time. More specifically this implies the maximization of cumulative reward in the long run and not an immediate reward. This notion of cumulative reward is described by a return as defined in Equation (2.6). $\gamma \in [0, 1]$ is the discount rate that enables the notion of a return to be utilized for both episodic and continuing tasks. Whereas episodic tasks are finite in time, continuing tasks are infinite, hence $T = \infty$. For $\gamma = 0$ the agent is only interested in the immediate reward. The agent is then referred to as myopic.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + ... + \gamma^{T-t-1} R_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{2.6}$$

Policies and Value Functions

The agent's behavioral strategy is called *policy* and is defined as in Equation (2.7). Value functions define a common understanding of how good the observed state is, or how good it is to take a specific action given an observed state. As the goal of the agent is to maximize a cumulative reward, the goodness is defined in terms of expected return. The rewards that the agent receives in the future depends on the its policy. Therefore the value functions are defined according to an policy π. Three value functions are defined. A *state-value*, *action-value* and an *advantage* function, by Equation (2.8), Equation (2.9) and Equation (2.10), respectively.

$$\pi(a|s) \doteq \Pr\{A_t = a|S_t = s\} \tag{2.7}$$

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s\right] \tag{2.8}$$

[1] The more general framework of POMDPs can be used to formulate partial observability.

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s, A_t = a\right] \tag{2.9}$$

$$a_\pi(s,a) \doteq q_\pi(s,a) - v_\pi(s) \tag{2.10}$$

The state-value function expresses the expected return by following the policy π starting at state s. Similarly, the action-value function defines the expected return by following policy π and taking action a at the initial starting state s. Last but not least, the advantage function expresses the additional expected return by taking action a at starting state s, compared to the expected return when starting at state s.

In the context of finite MDPs a policy is considered to be better if its expected return is greater than that of another policy for all states. The optimal policy π_* is the set of policies, which are equally better than all other policies. Accordingly, they share the same optimal state-value and optimal action-value functions, as defined in Equation (2.11) and Equation (2.12), respectively.

$$v_*(s) \doteq \max_\pi v_\pi(s) \tag{2.11}$$

$$\begin{aligned} q_*(s,a) &\doteq \max_\pi q_\pi(s,a) \\ &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] \end{aligned} \tag{2.12}$$

Bellman Equations

The value functions can be reformulated into a recursive form, also called Bellman equations. Equation (2.13) formulates the Bellman equation for v_π. This recursive property allows for the values of a state to be expressed as values of its possible successor states, forming the basis for the frameworks presented in the subsequent sections.

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\big[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\big] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma v_\pi(s')\big] \end{aligned} \tag{2.13}$$

2.1.2. Common Approaches

In RL three fundamental classes of solution methods are distinguished, Dynamic Programming (DP), Monte Carlo (MC) methods and Temporal Difference (TD) learning, with each their strengths and weaknesses.

Dynamic Programming

DP are algorithms that compute an optimal policy and value functions utilizing complete knowledge of the environment as a MDP. In this section the use of finite MDPs is continued, although the DP framework can also be applied to continuous state and action spaces. Close to all RL methods, including DP methods, can be viewed as Generalized Policy Iteration (GPI), which defines the interaction of policy evaluation and policy improvement. Policy evaluation stands for the computation of the value functions for a given policy. An iterative way of conducting policy evaluation by means of the Bellman equation Equation (2.13) is presented in Equation (2.14). Policy improvement refers to finding a better policy given the value function of the original policy. The improved policy $\pi'(s)$ is given by Equation (2.15).

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma v_k(s')\big] \end{aligned} \tag{2.14}$$

$$\begin{aligned}\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]\end{aligned} \tag{2.15}$$

Monte Carlo Methods
In contrary to DP, MC methods learn an optimal policy and value functions from sample sequences of states, actions, and rewards and therefore do not require any knowledge of the environment. Another distinction of MC methods from DP and TD learning, is that they do not bootstrap [2], making them less vulnerable to violations of the Markov property. As introduced for DP methods, the GPI principle is utilized. For policy improvement, the previous approach as defined by Equation (2.15) can be used. In the case that no model of the environment is utilized, policy evaluation by means of estimation of a state-value function does not suffice. Rather, policy evaluation through the estimation of the action-value function is necessary. Equation (2.16) defines the estimate of the action-value function based on the trajectory of one episode, where $\mathbb{1}$ is the binary indicator function.

$$q_\pi(s,a) \approx Q(s,a) = \frac{\sum_{t=1}^{T} \left[\mathbb{1}_{S_t=s, A_t=a} \sum_{k=0}^{T-t-1} \gamma^k R_{t+1+k} \right]}{\sum_{t=1}^{T} \mathbb{1}_{S_t=s, A_t=a}} \tag{2.16}$$

Temporal Difference Learning
In the context of GPI, DP, MC and TD learning differ mainly in their policy evaluation process. In essence TD learning combines the idea of MC methods of learning from samples, with the bootstrapping property of DP methods. For the concept of a general update rule, as defined by Equation (2.17), the update rule for an estimate of the state-value function using MC and TD learning methods are represented by Equation (2.18) and Equation (2.19) respectively, with step-size α.

$$\text{New Estimate} \leftarrow \text{Old Estimate} + \text{Step Size}\big[\text{Target} - \text{Old Estimate}\big] \tag{2.17}$$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{2.18}$$

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2.19}$$

Recall, the definition of the state-value function in Equation (2.20). Generally speaking, whereas MC methods utilize an estimate of the first row of Equation (2.20) as a target, DP and TD learning methods utilize an estimate of the last row. For MC methods a sample return, computed after the termination of an episode, is utilized in place of the expected return in the first row of Equation (2.20). For DP methods, the expectation, as in the last row of Equation (2.20), can be computed as a model of the environment is known. However, as $v_\pi(S_{t+1})$ is not known, the current estimate $V(S_{t+1})$ is utilized. Hence, they bootstrap. TD learning methods estimate the last row of Equation (2.20) through sampling of the expectation and through bootstrapping (the use of $V(S_{t+1})$ as estimate for $v_\pi(S_{t+1})$).

$$\begin{aligned}v_\pi(s) &\doteq \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_s = s]\end{aligned} \tag{2.20}$$

In the framework of TD learning, the error between the target and the old estimate, as in Equation (2.17) and Equation (2.19), is a quantity named TD error. The TD error, as defined in Equation (2.21), is used extensively throughout RL.

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2.21}$$

[2] Compute an estimate based on a previous estimate.

Synopsis
In contrary to DP, MC methods do not require complete knowledge of the environment. However, MC methods are not well-suited for applications with continuing tasks nor online implementations. Last but not least, TD learning requires no knowledge about the environment and is well-suited for online implementation due to its bootstrapping nature, but is generally more complex to analyze.

2.1.3. Dimensions

The approaches presented in Section 2.1.2 should not be viewed as individual methods but as a meaningful set of concepts utilized across methods. The large variety of methods in RL can be comprehensively portrayed by discussing dimensions along which they can vary.

Update Width and Depth
Two main dimensions are illustrated by Figure 2.2, as extracted from [100]. The dimensions differentiate the width and depth of the update procedure, utilized to improve the value function estimation. Three out of the four corners are represented by the ideas discussed in Section 2.1.2. The left edge encompasses methods that utilize sample observations, ranging from one-step TD learning updates to MC updates with return estimates at episode termination. In between these two corners are generalizations of TD learning that utilize reward samples from n sequential state-transitions. The horizontal edge differentiates whether the updates are conducted with sample estimates or expectation computations. Expectation computations require a distribution model, whereas sampling can be done from actual interaction or a sample model. In the upper right corner is DP as it utilizes expectation computations over one-step. The other extreme is exhaustive search, were expectation computations are conducted until episodic termination.

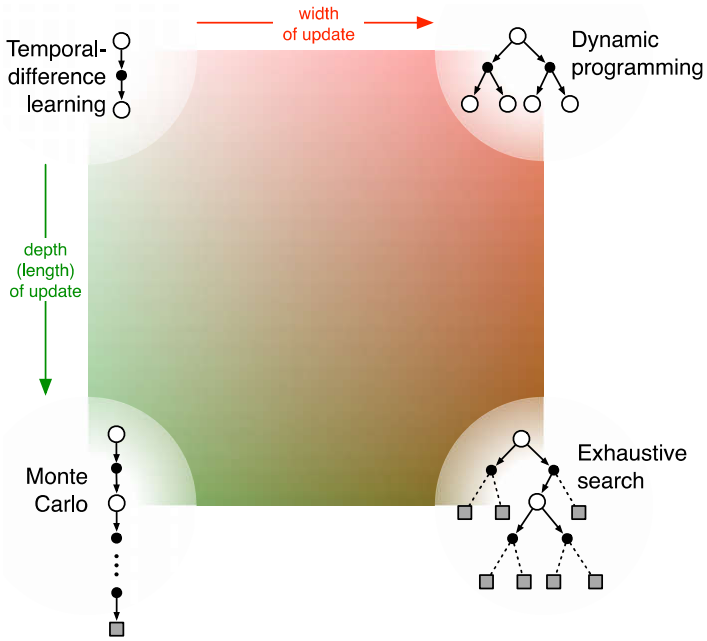

Figure 2.2: Representation of two main dimensions of RL methods [100]. The depth and width of the updates.

On-policy vs. Off-policy
A binary distinction is made between on-policy and off-policy approaches. In the former, the agent learns from the policy it is executing in the environment. In the latter, the agent learns from a policy different to the one it is executing. Off-policy learning allows for the agent's behavior to be decoupled from its learning process, providing more flexibility, especially concerning the exploration and exploitation trade-off [3]. Off-policy learning, however, suffers from high variance and slower convergence. Espe-

[3]The dilemma arises from incomplete knowledge. Sufficient information is necessary to make the best overall decisions. With exploitation, the agent takes advantage of the best option it is aware of. With exploration, the agent takes the risk to collect information about unknown options.

cially when combined with bootstrapping methods and function approximation, they make up the *deadly triad* [4].

Complete vs. Incomplete Environment Knowledge
Some approaches like DP require complete knowledge of the environment. Other approaches like MC methods can solely rely on samples. This distinction, however, is not necessarily binary. May methods only require partial knowledge of the environment.

Model-based vs. Model-free
Learning can be conducted not only from experience through interaction with the environment but also from interaction with a model of the environment. Methods that utilize a model of the environment are named model-based methods, in contrary to model-free methods, which don't use a model of the environment. Models can be known or learned [5]. Many model-based methods focus on learning the environment to subsequently used it for planning. Other methods, such as Dyna [102], utilize experience from both the model and environment for its value function estimation.

Function Approximation
Function approximation is an important dimension that ranges from tabular methods for discrete state and action spaces to methods with linear or nonlinear parameterized approximations. The framework of discrete state and action spaces was utilized in the previous sections to facilitate the discussion of the fundamental methods. However, tabular methods have limited applicability because of the *curse of dimensionality* [6].

2.2. Reinforcement Learning in Continuous Spaces

Many real-world applications, including the PH-LAB research aircraft, have continuous state and action spaces. However, due to the complexity of the RL problem in continuous spaces, traditional methods and discussions, including the previous sections, are focused on MDPs with small, finite state and action spaces. Nonetheless, their understanding is vital for the discussion of the RL problem in continuous spaces. As indicated in Section 2.1.3, function approximation is key for the applicability of RL to artificial intelligence or to large engineering applications. Due to the curse of dimensionality, tabular methods are not only restricted by memory, but by time and data. The key idea of approximate methods is to *generalize* the information from experienced scenarios, to similar unexplored scenarios.

This section starts by introducing several approaches to function approximation. Next, RL methodologies used for solving RL problems in continuous spaces are discussed in Section 2.2.2. These methods are categorized by which structure (model, value or policy) is approximated. Conclusively, the methods most promising to an application on the PH-LAB research aircraft are compiled in Section 2.2.3.

2.2.1. Function Approximation

The first distinction made is whether the function approximation method is parametric or non-parametric. Non-parametric methods such as memory-based methods have the advantage over parametric methods of not limiting approximations to pre-specified functional forms. However, generally, their computational speed degrades as the memory accumulates. An example of memory-based methods is nearest neighbors. The required computation time of nearest neighbors for a large amount of memory makes it not suitable for real-world applications such as the PH-LAB research aircraft. Therefore, the subsequent focus is put on parametric approximation methods.

As the name suggests, for parametric methods, the approximate function is defined by a set of tunable parameters. A distinction is made between linear and nonlinear parametric approximation. Linear methods are linear in the parameters. Hence, they form a linear combination of extracted features. These features can be constructed with polynomials, the Fourier series or discretization methods such as tile coding. A discussion on feature construction is presented in [14]. Generally speaking, linear

[4] The deadly triad refers to the instability and divergent behavior that methods which combine, off-policy learning, bootstrapping and function approximation, exhibit.
[5] Model learning is also referred to as system identification in adaptive control literature.
[6] Computational intractability due to an exponential increase of states with an increase in state variables.

methods are better understood than nonlinear function approximators and in the context of RL, some convergence guarantees have been proven under various additional assumptions [19][20][103]. However, for the special cases where convergence is guaranteed, the solution is only the optimal linear combination of the constructed features. Hence, the solution is constrained to the set of functions that are expressible by the constructed functional form. Furthermore, the main disadvantage of linear methods compared to nonlinear methods is the requirement for informative features, which are constructed in advance and required domain knowledge. Although less theoretical guarantees are given, nonlinear function approximators, such as Artificial Neural Networks (ANNs), are extensively used and have shown good empirical results. In fact, in recent years these methods have become very popular under their own branch named Deep Reinforcement Learning (DRL) [5]. Conclusively, the subsequent focus is towards nonlinear parametric function approximation with ANNs.

Last but not least, there are different methods for updating the parameters. Some RL algorithms support closed-form optimization of parameters, such as Least Squares Temporal Difference (LSTD) learning [30], for a set of observations. These optimization methods, however, are not applicable to nonlinear algorithms, such as Q-learning [114], nor to nonlinear function approximators, such as ANNs. Bayesian methods are interesting because they provide a framework for the agent's behavior that stimulates the exploration of parts of the model of the environment that have high uncertainty [69]. However, Bayesian methods are suitable for stationary functions. Hence, generally, they are suitable for learning a model of the environment, but not the value of a changing policy. Nevertheless, there are several methods which are compatible with both linear and nonlinear function approximators and can be utilized for learning a model, value function or policy. A distinction is made between Gradient Descent (GD) and gradient-free optimization methods. Gradient-free methods, such as evolutionary algorithms [99], particle swarm optimization [36] and cross-entropy optimization [75], are especially useful when the function to be optimized is not differentiable. These methods generally compute a population of solutions. The subsequent population is derived from the previous population based on a measure of fitness. The main disadvantage is the computational cost of the measure of fitness. On the other hand, GD can be applied in batches or sample by sample. The latter is named Stochastic Gradient Descent (SGD) and is especially suitable for online RL, as it can be utilized online for non-stationary targets, for example when the policy changes after an update. As the name implies, GD requires differentiability of the function that is optimized. As GD can be implemented online, it is the most suitable set of parameter optimization methods for the PH-LAB research aircraft. The differentiability requirement is not a problem as ANNs are inherently differentiable. Figure 2.3 summarizes the choices made in the discussion on the functional approximation methods. It should be noted that the illustrated distinctions are not mutually exclusive.

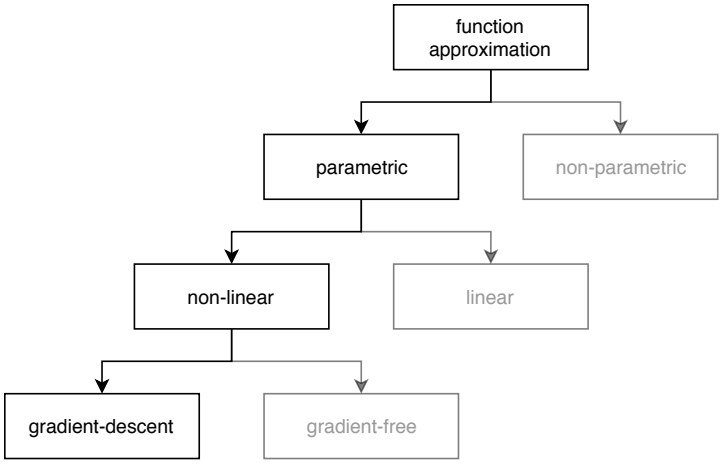

Figure 2.3: Selection tree of methods for function approximation and their optimization.

2.2.2. Methodologies for Solving Continuous MDP

In the context of the PH-LAB research aircraft, the goal is to approximate the optimal policy. As observable from Equation (2.12), the optimal policy depends on the optimal value functions, which in turn

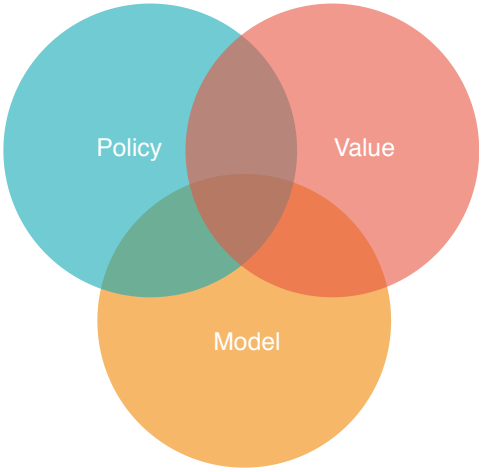

Figure 2.4: Venn diagram of Reinforcement Learning elements that are approximated.

depend on the model of the environment, assuming the environment is unknown. All of these three elements (policy, value functions or model) can be approximated, giving rise to three general methodologies. As illustrated in Figure 2.4 these methodologies are not mutually exclusive and generally RL algorithms consist out of combinations of these methodologies.

Model Approximation
Model approximation amounts to learning an approximation of the state-transition and reward processes as defined in Section 2.1.1. In some cases, especially in engineering applications, the reward process is considered to be known. Although learning a model of the environment is not trivial, it is generally easier than optimizing a policy or learning its value. Subsequently, the learned model of the environment can be used to determine a value function, for example by means of the DP ideas introduced in Section 2.1.2. However, this approach becomes intractable for continuous and therefore infinitely large state spaces. Alternatively, when combined with an approximation of the value function or policy, as elaborated in the subsequent sections, approximate models can be used to generate samples. However, in some cases, the learned policy may be worse than a policy learned directly from samples utilized for learning the model. A survey on model-learning algorithms is presented in [64].

Value Approximation
As the name suggests, value approximation amounts to learning an approximate value function from interaction. Additional to the distinction between on and off-policy, as described in Section 2.1.3, some algorithms can be applied offline, online or both. Common on-policy algorithms are Sarsa [100], Expected-Sarsa [109], Least Squares Temporal Difference (LSTD) [30][13], Least Squares Policy Evaluation (LSPE) [60] and Least Squares Policy Iteration (LSPI) [43]. Most commonly known is the off-policy Q-learning algorithm as originally introduced in [114] and its many variations, such as Bayesian Q-learning [21], Perseus [96], fitted Q-iteration [74][4] and Delayed Q-learning [98]. In [105] and [106] it was shown that Q-learning suffers from an overestimation bias, which can lead to divergence and the Double Q-iteration algorithm was proposed to mitigate this issue. The main disadvantage of these algorithms with respect to an application in the PH-LAB research aircraft is their non-trivial extension to continuous action spaces. Even though a value estimate is easily constructed for continuous actions, finding the maximizing action, as in Equation (2.12), in an infinite set is non-trivial. Despite its general limitation to discrete action spaces, Q-learning algorithms are still actively researched and have contributed to major breakthroughs in modern general artificial intelligence research, such as the Deep Q-Network (DQN) [56] that presented performance superior to humans in a set of 49 games.

Policy Approximation
Methods that utilize an approximation of the policy can naturally handle continuous state and action spaces, in contrary to the aforementioned value approximation methods. Additionally, they are able to asymptotically approach a deterministic policy. Thus, policy approximation methods directly update the policy towards the optimal policy and are referred to as direct policy-search or actor-only algorithms.

Algorithms that approximate the value function additionally to the policy, are named actor-critic algorithms. By this terminology, the aforementioned value approximation methods can be referred to as critic-only methods.

One way to optimize the policy is by gradient-free policy search, e.g. by means of evolutionary strategies [99]. However, due to the inherent large variance, policy search is generally not suitable for large problems. Most commonly, policy-gradient methods are utilized, whose theoretical foundation is the policy-gradient theorem as first derived by [54]. They update the policy in the ascending direction of the cumulative expected value, as in Equation (2.8). Since in general the cumulative expected value, nor its gradient with respect to the actor is known, an estimate is employed.

The policy-gradient theorem provides a notion of this gradient, which is independent of the transition model for stochastic policies, also called stochastic policy gradient. A derivation is provided in [115]. Combined with a MC estimation of the return, as in Equation (2.6), it establishes the fundamental policy-gradient algorithm REINFORCE. The variance of the gradient estimate in the REINFORCE algorithm can be quite high, thus leading to slow learning. Therefore, the use of a baseline to minimize variance without adding bias is common practice and extensively studied [116] [101] [32] [67]. The MC estimate of the REINFORCE algorithm is not only of high variance but is also inconvenient for online implementation or for continuing tasks. This inconvenience is mitigated by actor-critic methods which use TD learning in conjunction with a bootstrapping critic, making them most suitable for an application on the PH-LAB research aircraft.

2.2.3. Synopsis

As discussed in Section 2.2.2 it is intractable to analytically determine a good policy directly from an approximate model. Approximating the state-action value function facilitates this process, as for each state, there is at least one value maximizing action.[7] For a continuous action space, however, as is the case for the PH-LAB research aircraft, determining the greedy action is non-trivial and expansive. Therefore, methods that approximate a policy are most suitable for the PH-LAB research aircraft as they inherently support continuous state and action spaces. The design on an adaptive flight controller naturally calls for an online learning process. Furthermore, the flight control task in the PH-LAB aircraft is by nature a continuing task and not episodic. As a result, a policy-gradient actor-critic approach, in conjunction with nonlinear function approximation with ANNs, is the most suitable for an application in the PH-LAB aircraft.

Originally, actor-critic methods were among the earliest to be studied in RL [117] [10], until being superseded by action-value methods in the 1990s. However, in recent years, a lot of attention returned to actor-critic methods [34] and many of today's state-of-the-art algorithms are actor-critic approaches [5]. Actor-critic methods have not only been studied across different time intervals but across different fields of study. Significant parallels are observable in the related field of Approximate Dynamic Programming (ADP) [70] [86]. Although RL and ADP are often thought of particular names for the same research field, a distinction is made in the type of problems they are applied to. ADP utilizes an engineering's perspective with an accompanying distinct notation and is commonly applied to control tasks [49]. Furthermore, many ADP algorithms assume knowledge of the state-transition process, reward process or both, or establish models of these processes. In the latter, a common additional requirement is that these models are differentiable.

The distinction of the research field is maintained in the subsequent sections. First, actor-critic approaches in the field of ADP are discussed in Section 2.3.1, followed by a review of their applications to flight control in Section 2.3.2. In Section 2.4 state-of-the-art actor-critic approaches in RL research, as commonly applied to general artificial intelligence research, is reviewed.

2.3. Approximate Dynamic Programming

This section starts by introducing the branch of Adaptive Critic Designs (ACDs) in Section 2.3.1. Subsequently, their application in the flight control domain is reviewed in Section 2.3.2.

2.3.1. Adaptive Critic Designs

The class within ADP which utilizes actor-critic approaches in commonly referred to as ACDs. The original ACDs as introduced in [55] utilized ANNs as memory structures, but as discussed in Section 2.2.1

[7] Also referred to as greedy actions.

other approximation methods can be applied. ACDs are categorized in three main groups: Heuristic Dynamic Programming (HDP), Dual Heuristic Programming (DHP) and Globalized Dual Heuristic Programming (GDHP). Furthermore, there are action dependent versions, which are named accordingly: Action Dependent Heuristic Dynamic Programming (ADHDP), Action Dependent Dual Heuristic Programming (ADDHP) and Action Dependent Globalized Dual Heuristic Programming (ADGDHP). In [71] all types are treated in detail. In general they can be distinguished by two attributes: The input to and output from the critic and the requirement of a state-transition model, also referred to as plant model, for the training process. Table 2.1 summarizes the different ACDs structures with their distinct attributes.

Table 2.1: Summary of distinct attributes of different Adaptive Critic Designs structures.

Structure	**Critic Topology**		**Plant Model Requirement**	
	Input	Output	Actor	Critic
HDP	s	v	yes	no
ADHDP	$\begin{bmatrix} s & a \end{bmatrix}$	q	no	no
DHP	s	$\frac{\partial v}{\partial s}$	yes	yes
ADDHP	$\begin{bmatrix} s & a \end{bmatrix}$	$\begin{bmatrix} \frac{\partial q}{\partial s} & \frac{\partial q}{\partial a} \end{bmatrix}$	no	yes
GDHP	s	$\begin{bmatrix} v & \frac{\partial v}{\partial s} \end{bmatrix}$	yes	yes
ADGDHP	$\begin{bmatrix} s & a \end{bmatrix}$	$\begin{bmatrix} q & \frac{\partial q}{\partial s} & \frac{\partial q}{\partial a} \end{bmatrix}$	no	yes

Whereas the actor is a representation of the policy, hence a mapping of state to action, the mapping of the critic depends on the ACDs structure. Typically, the critic's input consists of the state. For the action dependent versions, the critic also accepts the action from the actor as input, additionally to the state. Accordingly, for action dependent versions the critic is a representation of the action-value function instead of the state-value function.

In HDP the critic is a direct mapping of state to state-value function and therefore doesn't require a plant model for its update. In contrary, the update of the actor requires the derivative of the state-value function with respect to the action $\frac{\partial v}{\partial a} = \frac{\partial v}{\partial s}\frac{\partial s}{\partial a}$. Hence, backpropagation is conducted through the critic and the required plant model. ADHDP allows for the gradient $\frac{\partial v}{\partial a}$ to be directly extracted from the critic and therefore does not require a plant model. In DHP and GDHP the critic represents the derivative of the state-value function with respect to the state $\lambda = \frac{\partial v}{\partial s}$. Consequently the identity as elaborated in Section 3.2.2 by Equation (3.14) required for the update processes, needs plant model derivatives $\frac{\partial s}{\partial s}$, $\frac{\partial s}{\partial a}$ and reward function derivative $\frac{\partial r}{\partial s}$. ADDHP and ADGDHP alleviate the plant model requirements for the update process of the actor, as the required derivatives can be directly extracted from the critic. However, the update process of the critic still requires a plant model and reward function derivatives.

Different studies [71] [110] have shown that the ACDs structures DHP and GDHP exceed HDP in precision and success rate. According to [85] this performance difference is mainly attributed to the critic's representation of the value function derivatives. Furthermore, the computationally more complex GDHP has not displayed distinct advantages over DHP. Although, the action dependent versions, which alleviate the model dependency have been widely studied [23] [65], HDP has shown superior performance to its action dependent counterpart ADHDP when applied to flight control [71] [107]. ACDs have been extensively applied to online learning control tasks, but commonly rely on two distinct learning phases [111] [27] [23]. An offline and an online learning phase. In most cases, the required plant model is unknown and has to be identified. The identification of the plant model is nontrivial and therefore requires an initial offline learning phase. Unfortunately, the identification on a complete plant model is still accompanied by several drawbacks. First, the offline plant model identification process relies on the availability of representative simulation models. Furthermore, complex plant models may be too computationally expansive to adapt to changes in the environment during online operation.

In [85] and [65] methods were proposed, which replace the plant model by utilizing a time series of the critic's inputs and outputs. Although successful, the proposed modifications only relieve the offline learning phase of the action dependent structures. Especially for DHP and GDHP an accurate plant model remains vital, as both the update process of the actor and critic depend on it, as illustrated in Table 2.1.

In an effort to reduce the dependence of ACDs to a priori plant information, [124] and [124] proposed the use of a linear, time-varying incremental model, as originally used in adaptive control design [72] [87] [2], instead of a complete plant model. The proposed methods named, Incremental Heuristic Dynamic Programming (IHDP) and Incremental Dual Heuristic Programming (IDHP), successfully eliminated the need for an offline training phase and improved the online learning efficiency for simple dynamic systems. However, the proposed methods have yet to be applied to and studied on complex, high-dimensional models and real systems.

2.3.2. Adaptive Critic Designs for Flight Control

In Section 2.3.1 actor-critic methods as established in the research field of ADP were introduced and discussed. As ADP has its roots in engineering, many of the introduced methods have been applied to (flight) control tasks. In this section, the major studies on the application of ACDs to flight control are reviewed, allowing for common limitations to be exposed and common procedures to be identified.

Adaptive-Critic-Based Neural Networks for Aircraft Optimal Control

One of the first applications of actor-critic approaches with ANNs to aircraft flight controls is presented in [6]. The author proposes a DHP framework as discussed in Section 2.3. The research, however, is limited to a single control along the longitudinal axis of a Linear Time Invariant (LTI) model. Conclusively, the author demonstrates that the state histories generated by the learning controller are close to identical to the optimal trajectory, acquired by solving the Riccati equation for a Linear Quadratic Regulator (LQR) based state-feedback controller.

Adaptive Critic Designs

In [71] a detailed review of the ACDs families, as discussed in Section 2.3, is presented and modifications are proposed to establish a unified training approach for all ACDs. One of the experimental studies in the paper included the application to an auto-lander for a simple, linearized model of a commercial aircraft, with the goal of landing the aircraft within a specified touchdown region, including constraints to speed and pitch, deterministic wind shear and stochastic wind gusts. The landing performance was compared between a PID controller, ADHDP, HDP, DHP and GDHP, with the ACDs showing improving performance in that order. The controller computed the required elevator angle command, based on the aircraft's states and desired altitude and vertical speed, as supplied by an Instrument Landing System (ILS).

Helicopter Trimming and Tracking Control Using Direct Neural Dynamic Programming

According to the authors in [23], this is one of the first systematic application and evaluation of an ADP methodology on a complex, continuous, nonlinear, Multiple Inputs Multiple Outputs (MIMO) system with uncertainty. In this study, the ACDs approach named Direct Neural Dynamic Programming (DNDP) is applied to stability and tracking control of an Apache helicopter simulation model. Despite the different naming, DNDP can be classified as a ADHDP structure, as introduced in Section 2.3. Consequently, this method theoretically does not require a plant model for the training processes of the actor nor critic, as discussed in Section 2.3. As the name suggests, ANNs are utilized as approximators. The high fidelity Apache helicopter simulation model, named FLYRT [42], is run at 50 Hz. For stability control five flight conditions with different velocities are analyzed, followed by a tracking control analysis through several flight maneuvers. In both cases, simulations are conducted for both clear air and in the presence of turbulence, modeled by an industrial Dryden model. Subsequently, it is demonstrated how the proposed framework is capable of controlling the helicopter despite an induced actuator failure, whereas a PID controller becomes unstable.

This study demonstrates that the proposed DNDP framework is capable of successfully controlling a complex, continuous MIMO system, online and in the presence of multi-axis coupling. According to the authors, the introduction of an isolated network for trimming and a cascaded actor topology, as illustrated in Figure 2.5, are critical to a successful implementation. The so-called trim network provides the nominal trim control positions for specific operating conditions and is trained offline, in advance and independent from the actor and critic networks. It should be noted that for the framework to successfully learn to perform the maneuvers, a large number of trials is necessary. Consequently, initial training needs to be conducted offline (i.e. in simulation), where failures are tolerable. Only then, the trained controller can be implemented in the real system with limited authority online training.

Hence, due to the offline learning phase of the controller and the training process of the trim network, an accurate plant model of the system is still required a priori.

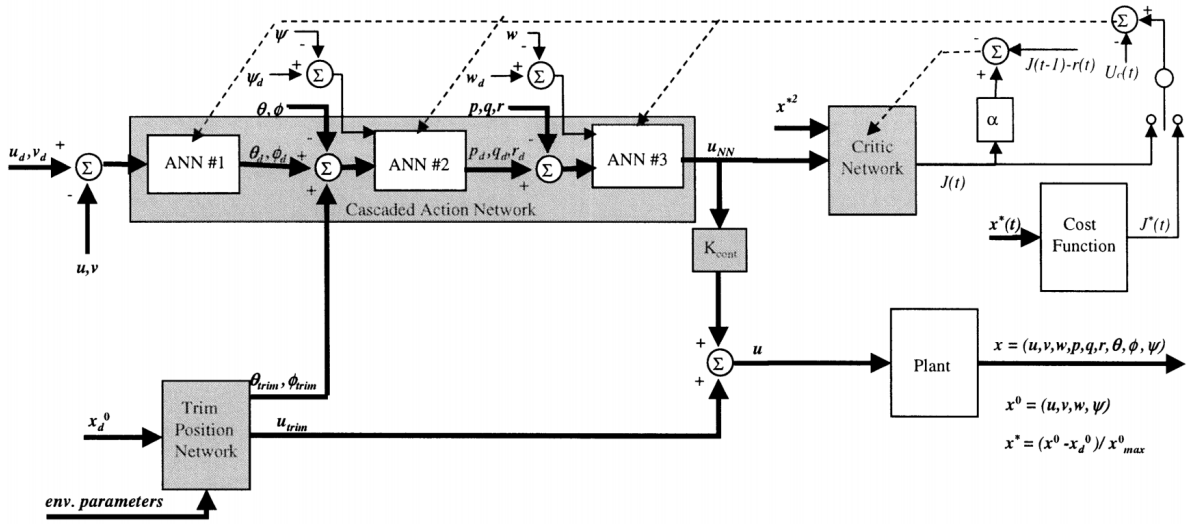

Figure 2.5: DNDP-based helicopter controller integrating three cascaded artificial neural networks in one action network, a trim network, and a critic network, as presented in [23].

Online Adaptive Critic Flight Control

In [27] a DHP framework with ANNs as parametric structures is utilized for the control of a six-degree-of-freedom simulation of a business jet aircraft over its full operating envelope. There are two distinct learning phases. In the offline learning phase networks are trained by the idea that the gradients should correspond to linear gain matrices at specific operating points. According to the authors, this observation, derived from multivariable control theory, allows for satisfactory safety and performance baselines to be met a priori, as knowledge from well-known gain-scheduled linear controllers is incorporated [84] [26]. Similarly to [23], an additional ANN is pretrained offline to approximate the vehicle's trim map [25] and frozen during online operation. During the online training phase, the controller learns to optimize the performance over time by accounting for differences between the actual and assumed dynamic models, such as nonlinear effects that prevail in large coupled motions. Furthermore, the authors conduct numerical experiments to show that prior knowledge is preserved during online learning.

Subsequently, experimental results are presented for four different cases: a coupled maneuver, a large-angle maneuver, multiple control failures, and parameter variations. The performance between the adaptive online learning controller is compared to a static controller resulting from the offline training phase. In all cases, the adaptive controller outperforms the static controller. In contrary to the adaptive controller, the static controller is not able to recover from a stall during the large angle maneuver. However, the authors emphasize that the stall region is not accurately represented by the aircraft's model. Conclusively, the proposed adaptive controller is able to improve its performance compared to its initial pretrained configuration during operation and in the presence of unexpected control failures, unmodeled dynamics and parameter variations.

Continuous Adaptive Critic Flight Control Aided with Approximated Plant Dynamics

As elaborated in Section 2.3 the ADHDP framework does not require a plant model, in contrary to its affiliated framework HDP. In [107] a comparison between the ADHDP and HDP is presented through the design and evaluation of a longitudinal flight controller for a model of a F-16 military jet. The longitudinal model has two control inputs, which are controlled individually. The pitch attitude and airspeed are controlled by the elevator deflection and the throttle setting, respectively. The experiments are divided into two phases. In the first phase, the frameworks are evaluated based on their capability of controlling the aircraft after offline training. Subsequently, the plant's dynamics are modified and the frameworks are compared based on their ability to adapt to the modified plant through online training. In the offline phase, the HDP framework attains a higher success ratio than ADHDP. Although both pretrained HDP

and ADHDP frameworks are able to cope with slight modifications to the plant's dynamics by default, HDP shows superior adaptability during online learning. Last but not least, the ADHDP framework is less sensitive to Gaussian measurement noise, but can only be used in a narrower range of initial flight conditions compared to HDP.

Online Reinforcement Learning Control for Aerospace Systems

The aforementioned publications, such as [27] [23], have shown that extended forms of ACDs can be successfully applied to flight control of complex dynamic systems. However, a common limitation is that a representative simulation model of the real environment is required a priori for the offline learning phase. Unfortunately, for many aerospace applications, no accurate system model is available, nor readily identifiable [83][51].

With the purpose of designing a controller that is able to adapt online without a priori knowledge of the system nor offline learning of the system model, recent studies proposed a redesign of the HDP and DHP framework based on the incremental control technique [123] [124]. The inspiration for the incremental approach originated from its successful application in adaptive controllers such as Incremental Backstepping (IBS) [2], Incremental Nonlinear Dynamic Inversion (INDI) [87] [90] and incremental adaptive sliding mode control [72]. The idea is to replace the plant model by a local, linear, time-varying incremental model approximation of the system. Assuming a sufficiently high sampling rate, the incremental model is identified solely online during operation through a Recursive Least Squares (RLS) estimator.

The new frameworks called IHDP and IDHP are introduced in [123] and [124] respectively. In both cases, the authors applied it to an angle of attack tracking control task on a simple, yet nonlinear, short period missile model [37]. In [123] a comparison is presented between HDP and IHDP. Conclusively, IHDP provides superior control precision, accelerated online learning and support for a wider range of initial conditions. In [124] a comparison is conducted between DHP and IDHP. A fault-tolerant control task experiment is conducted additionally to the tracking control task experiment. Similar to [123], the incremental framework IDHP shows superior control precision, adaptability and robustness, when compared to its affiliated DHP framework. Furthermore, in the presence of a sudden change to the system's dynamics, the IDHP framework is able to adapt and successfully regain control of the system, in contrary to DHP. Conclusively, the novel incremental frameworks are very attractive for applications without a priori knowledge of the system. However, their extension to larger, more complex models and real systems remains to be studied.

2.4. Deep Reinforcement Learning Actor-Critic Algorithms

As previously mentioned, the recent advent of deep learning has led to a significant acceleration in machine learning research, including RL. Deep learning is making it possible to scale RL to previously intractable complex problems, such as learning policies for robots directly from visual observations [45] [46]. Actor-critic approaches continue to play an exciting role in Deep Reinforcement Learning (DRL) research. In this section, some of the major contributions to research on policy gradient actor-critic approaches are reviewed.

2.4.1. Deterministic Policy Gradient Algorithms

Policies, as utilized in the REINFORCE algorithm, are commonly stochastic. In [88] an efficient approach for estimating the policy gradients, named Deterministic Policy Gradient (DPG), is presented. DPG is an extension of the standard policy gradient theorem for stochastic policies, as introduced in Section 2.2.2, to deterministic policies. In contrary to the stochastic policy gradient, which integrates over the state and action spaces, the DPG is the expected gradient of the action-value function, which integrates only over the state space. As a result, the DPG is more efficient to estimate than the stochastic policy gradient. Subsequently, the authors proposed an off-policy actor-critic approach capable of learning a deterministic target policy from an exploratory behavior policy. Recall the discussion on off-policy learning in Section 2.1.3. The authors utilized tile-coding and linear function approximators to ensure an unbiased policy gradient. DPG approaches compared to stochastic policy gradient approaches, have shown superior performance in standard benchmark RL problems, especially in high dimensions tasks.

2.4.2. Continuous Control with Deep Reinforcement Learning

In [47], ideas from the value-approximation algorithm DQN [56] and DPG [88] are combined to form an model-free, off-policy, actor-critic algorithm for continuous spaces, named Deep Deterministic Policy Gradients (DDPG). As elaborated in Section 2.1.3, combinations of off-policy learning, bootstrapping and function approximation, such as ANNs, lead to unstable and divergent learning behavior [100]. This fundamental instability problem was addressed in the DQN algorithm through the use of two techniques, which are also applied in DDPG: experience replay [48] and target networks. Experience replay enables the agent to learn from previously observed data by randomly sampling from a cyclic buffer of stored transitions in the form of $(S_t, A_t, R_{t+1}, S_{t+1})$. Subsequent research already proposed modified versions of experience replay, such as Prioritized Experience Replay (PER) [78] and Hindsight Experience Replay (HER) [3]. The second technique, first introduced in [56], utilizes so called target networks that are initially copies of the networks enacting the policy, but are kept frozen. Subsequently, these target networks are utilized for the computation of the target in the Temporal Difference (TD) error, as in Equation (2.17) and Equation (2.21). In contrary to the DQN algorithm, DDPG doesn't periodically update the targets networks by copying, but utilizes a "soft" update, as defined by Equation (2.22), where $\mathbf{w}$, $\mathbf{w}'$ and τ represent the network's weights, the target network's weights and a scalar, respectively. The use of target networks increases the stability of the training process by mitigating short-term oscillations.

$$\mathbf{w}' = \tau\mathbf{w} + [1 - \tau]\mathbf{w}' \tag{2.22}$$

Furthermore, the authors utilize batch normalization to account for the fact that observations commonly have different physical units, which disturbs the agent's learning process. Last but not least, the behavior policy is constructed by adding noise to the actor policy.

Subsequently, the authors applied the DDPG algorithm to more than 20 simulated physics tasks of varying difficulty in the MuJoCo environment [68]. The model-free algorithm was able to attain performance equivalent to that of a planning algorithm with full access to the dynamics of the environments. Although DDPG requires a lot of interaction with the environment to learn a good policy, it requires substantially fewer steps than DQN.

Subsequent publications have proposed modifications to improve DDPG. In [29] an algorithm named Twin Delayed Deep Deterministic (TD3) is proposed to combat overestimation of value functions by using a pair of critics and delaying target updates. Distributed Distributional Deep Deterministic Policy Gradients (D4PG), an algorithm introduced in [9], introduces the use of PER and multiple distributed parallel actors, which feed their gathered experience into the same replay buffer. Similarly an algorithm named Multi-Agent Deep Deterministic Policy Gradients (MADDPG) [52] employs multiple agents. Instead of gathering experience in separate environments, the agents in MADDPG cooperate in the same environment.

2.4.3. Asynchronous Methods for Deep Reinforcement Learning

In [57] a framework is proposed that optimizes deep neural network controllers through asynchronous gradient descent. The authors apply this framework to four RL algorithms. One of the resulting algorithms, named Asynchronous Advantage Actor-Critic (A3C), is one of the most prevalent RL algorithms in recent times. A3C combines two techniques, which are already implied by its name: advantage updates and asynchronous gradient descent. In the former, the advantage, as in Equation (2.10), is estimated through Generalized Advantage Estimator (GAE) [80] and subsequently utilized for the policy gradient update of the actor. The advantage function can significantly reduce the variance of the policy gradient estimate. Recall from Section 2.1.1 that the advantage function expresses the potential benefit one would get for changing the policy. Hence, it is easier to learn that one action is more preferable than another than it is to learn the actual return from taking the action. In the latter, policy and value function networks are trained in parallel, through the use of several agents that act individually in their own copy of the environment. This asynchronous parallel learning increases learning stability and allows for more exploration. Although A3C surpassed the performance of previously state-of-the-art DRL algorithms at half the training time, studies have already proposed improvements.

Researchers from OpenAI have proposed a synchronous, deterministic version of A3C, named Advantage Actor-Critic (A2C), which waits for all parallel agents to complete their work before updating the global parameters [119]. The A2C approach, although unpublished, leads to more cohesive training which is potentially more efficient.

In [113] a novel algorithm named Actor-Critic with Experience Replay (ACER) is proposed. As the name suggests, ACER is an off-policy actor-critic algorithm with experience replay, with A3C as its foundation. The authors introduce some modifications such as a different estimation algorithm and an efficient form of Trust Region Policy Optimization (TRPO). The former is an off-policy return-based action-value function estimation algorithm, named Retrace [59], with the benefit of having some convergence guarantees and being data efficient. The latter is introduced as part of a different algorithm in Section 2.4.4.

2.4.4. Policy Optimization

A logical approach to improve training stability would be to avert parameter updates that significantly modify the current policy within one step. In that way, the chance of a catastrophically bad update is decreased. This idea is realized in the algorithm Trust Region Policy Optimization (TRPO) [79], by imposing a Kullback–Leibler (KL) divergence constraint between the current and proposed policy. In TRPO this is accomplished by optimizing a surrogate objective function, constrained through a quadratic approximation of the KL divergence, which is a measure of dissimilarity between two probability distributions. However, the implementation of TRPO is rather convoluted and computationally expansive. In [81] an algorithm, named Proximal Policy Optimization (PPO), is introduced that applies a similar constraint as in TRPO but is more simple and less computationally expensive. This is achieved through a clipped surrogate objective function. Whereas TRPO applies KL divergence constraints outside of the objective function, PPO clips the ratio of the previous and proposed policies, within the objective function. TRPO has shown equivalent or superior performance to other state-of-the-art DRL algorithms, such as ACER and A2C, on a collection of RL benchmark tasks such as Atari games and simulated robotic locomotion. PPO shows similar performance to TRPO, better sample complexity and is much simpler to implement.

2.5. Conclusion

In the first part of the literature survey, the fundamentals of RL were introduced. For the sake of simplicity, the introduction was conducted in discrete state and action spaces. However, as formulated in Chapter 1, the goal is continuous control. Consequently, RL approaches for continuous state and action spaces, were discussed in Section 2.2. In Section 2.2.1 parametric, nonlinear function approximation with gradient-based optimization was identified to be most applicable to the PH-LAB aircraft. Emphasis is put on function approximation with ANNs as it is well established and extensively applied in literature, as presented in Section 2.3.2 and Section 2.4. In Section 2.2.2 it was concluded that methods that approximate the policy are most suitable for the research objective as they inherently support continuous state and action spaces. In combination with value approximation, these methods demonstrate better learning performance and can be applied online by means of TD learning. Conclusively, the scope of literature to be reviewed is defined by actor-critic methods with nonlinear function approximation with ANNs. These methods are applied in separate research fields, ADP and DRL, as reviewed in Section 2.3 and Section 2.4.

As part of ADP, ACDs have a great record in flight control applications. As discussed in Section 2.3, the majority are model-based methods and have the common limitation that an initial offline training phase is required. Common in several flight control applications is the use of an additional trim network and a cascaded actor structure. In contrary to ADP, methods in the research field of DRL focus on general artificial intelligence problems, which are more versatile and complex. Therefore the publications as reviewed in Section 2.4, focus on reduction of variance and improvement of training stability, through methods such as experience replay, target networks, update constraints and parallel learning. The methods in DRL sometimes apply updates during interaction with the environment, making them theoretically online methods. However, these algorithms are trained in episodic fashion in a simulated environment where failure is acceptable. Consequently, in the context of an application to the PH-LAB, these algorithms can not be applied online. As a result, the development of adaptive flight controller with DRL methods would consist of offline training utilizing a very large amount of data that is representative of various failure cases and conditions. The idea is that the resulting agent would have learned to generalize to any change it may encounter during operation a priori. Although DRL methods may make it possible to learn such levels of intelligence, the ADP approach, with online learning capabilities is more in line with the development of an adaptive flight controller and the research objectives.

From Section 2.3 it was concluded that the DHP and GDHP frameworks achieve the best performance. As DHP demonstrates close to equivalent performance to GDHP with lesser framework complexity, DHP is the preferred framework in literature. The novel IDHP algorithm proposes a method that eliminates the requirement of an offline training phase. As a result, it is the only approach that allows for the development of an online, model-independent adaptive flight controller as stated in the research objective. Conclusively, the IDHP framework is selected as a baseline approach for this research. Nevertheless, novel ideas for improving learning performance, as introduced in DRL should be kept in mind during later stages of development. With the proposal of the baseline framework, the first research goal (G1), as listed in Section 1.1 is attained.

3

Preliminary Analysis

From the framework of Adaptive Critic Designs (ACDs), Incremental Dual Heuristic Programming (IDHP) has been identified, in the literature survey, as a promising approach for the purpose of this research. The general concept of ACDs is easily understood. However, the wide variety of challenges associated with its implementation are non-trivial. Understanding these is vital for a successful application of ACDs. There are three main goals of the preliminary analysis. First, is to get acquainted with the implementation of the IDHP framework. Second, is to set up a preliminary agent, to verify it and to take initial development decisions. Last but not least, is to asses the feasibility of the agent for flight control of the PH-LAB research aircraft. This chapter starts with a clear formulation of the flight control task that is analyzed in Section 3.1. In Section 3.2, a simplified version of the Dual Heuristic Programming (DHP) framework, named Model Dependent Dual Heuristic Programming (MDDHP), is derived. Subsequently, the MDDHP agent is extended to the IDHP framework, in Section 3.3. Section 3.4 presents and discusses the results generated through experiments. Last but not least, the chapter is concluded in Section 3.5.

3.1. Problem Formulation

A clear formulation of the problem to be solved and the environment, which the agent interacts with, is an important starting point. This section starts by presenting the plant dynamics of the environment. Next, the flight control task is formulated followed by the definition of the reward function. In Reinforcement Learning (RL) literature, it is common to define the plant dynamics and reward function, as properties of the environment, whose mechanics are hidden from the agent. However, as in the RL branch of ACDs and in this chapter, the reward function is not considered to be a hidden process of the environment, but a deliberately designed functional.

3.1.1. Plant

A discrete-time, deterministic, Linear Time Invariant (LTI) short period model was selected as the plant. The short period model was derived by simplifying and discretizing the symmetric motion model as presented in [58]. Its simplicity, due to its linearity and low dimensionality, facilitates the implementation and verification of the agent, but still provides a good approximation of the PH-LAB research aircraft. Additionally, the plant dynamics are differentiable. Although this is not necessary for the framework of IDHP, it is a requirement for the simplified framework of MDDHP treated in Section 3.2.

The short period model contains a two-dimensional state $\mathbf{s} = \begin{bmatrix} \alpha & q \end{bmatrix}^T$ and a one-dimensional action $\mathbf{a} = \begin{bmatrix} \delta_e \end{bmatrix}$, whereas the states are the angle of attack α and pitch rate q and the action is the elevator deflection δ_e, respectively. Equation (3.1) and Equation (3.2) define the state-transition function in their continuous and discrete-time forms, respectively. The values for the state matrix $\mathbf{A}$, the input matrix $\mathbf{B}$ and sample time Δt, as derived from [58], are presented in Table 3.1. The symmetric motion model and its parameters are presented in Appendix A. The partial derivatives of the discrete-time state transition function $f(\cdot)$ with respect to the state $\mathbf{s}_t$ and action $\mathbf{a}_t$ are defined by Equation (3.3) and Equation (3.4), respectively.

$$\dot{\mathbf{s}} = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{a} \tag{3.1}$$

$$\begin{aligned} \mathbf{s}_{t+1} &= f(\mathbf{s}_t, \mathbf{a}_t) \\ &= [\mathbf{A}\Delta t + \mathbf{I}]\mathbf{s}_t + \mathbf{B}\Delta t\mathbf{a}_t \\ &= \mathbf{F}\mathbf{s}_t + \mathbf{G}\mathbf{a}_t \end{aligned} \tag{3.2}$$

$$\begin{aligned} \frac{\partial f(\mathbf{s}_t, \mathbf{a}_t)}{\partial \mathbf{s}_t} &= \mathbf{A}\Delta t + \mathbf{I} \\ &= \mathbf{F} \end{aligned} \tag{3.3}$$

$$\begin{aligned} \frac{\partial f(\mathbf{s}_t, \mathbf{a}_t)}{\partial \mathbf{a}_t} &= \mathbf{B}\Delta t \\ &= \mathbf{G} \end{aligned} \tag{3.4}$$

Table 3.1: State matrix, input matrix and sample time for short period model of PH-LAB research aircraft [58].

Variable	Value	Unit
$\mathbf{A}$	$\begin{bmatrix} -7.391 \cdot 10^{-1} & 9.744 \cdot 10^{-1} \\ -1.472 & -1.567 \end{bmatrix}$	$\left[-\right], \left[\frac{1}{s}\right], \left[\frac{1}{s^2}\right]$
$\mathbf{B}$	$\begin{bmatrix} -8.935 \cdot 10^{-2} \\ -6.722 \end{bmatrix}$	$\left[\frac{1}{s}\right], \left[\frac{1}{s^2}\right]$
Δt	0.02	$[s]$

3.1.2. Flight Control Task

The flight control task is best described as a variable set-point tracking task. The goal is to find an optimal policy $\pi(\cdot)$ that maps the current state $\mathbf{s}_t$ and reference state $\mathbf{s}_t^R$ to an action $\mathbf{a}_t$, as defined in Equation (3.5), such that, a discounted sum of future rewards g_t, as in Equation (3.7), is maximized. The discount factor γ is a parameter of the agent. The reward r_t at time t, as provided by the reward function $r(\cdot)$ during a state-transition, is a function of the state to which the plant transitioned to $\mathbf{s}_{t+1}$ and the previous reference state $\mathbf{s}_t^R$, as defined by Equation (3.6).

$$\mathbf{a}_t = \pi(\mathbf{s}_t, \mathbf{s}_t^R) \tag{3.5}$$

$$r_{t+1} = r(\mathbf{s}_t^R, \mathbf{s}_{t+1}) \tag{3.6}$$

$$g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \tag{3.7}$$

For the preliminary analysis, a rate reference tracking control task is selected, as it is less complex than attitude reference tracking. The lower complexity originates from the more direct dynamic relation of the elevator deflection δ_e to the pitch rate q, than to angle of attack α, as can be observed from the state and input matrices in Table 3.1. This argument would also apply to the pitch angle θ when augmenting the state and input matrices. A sinusoidal function, with amplitude A and frequency f, was selected for its simplicity. Other functions types, such as doublets, are commonly used for system identification [39] and are interesting candidates for future development. The values for the signal parameters are presented in Table 3.2. No aliasing occurs as the sampling frequency $\frac{1}{\Delta t}$, as in Table 3.1, is sufficiently larger than the reference signal's fundamental frequency.

$$q^R = A\ sin(2\pi f t) \tag{3.8}$$

Table 3.2: Amplitude and frequency of sinusoidal pitch rate reference signal.

Variable	Value	Unit
A	5.0	$[deg]$
f	0.1	$[Hz]$

3.1.3. Reward Function

All the information the agent receives about its performance is contained in the reward function. Hence, in the context of optimal control, the optimality of the policy is strictly defined in terms of the reward function. Therefore, it is important that the designed reward function captures the requirements of the task at hand. Recalling that the task at hand is variable set-point tracking, the reward function is defined as the weighted, squared state tracking error, as presented in Equation (3.9). The Boolean selection matrix $\mathbf{P}$ is utilized to extract the intended states from the state $\mathbf{s}_t$ and $\mathbf{Q}$ is the symmetric weight matrix. Hence, for the state $\mathbf{s}_t = \begin{bmatrix}\alpha_t & q_t\end{bmatrix}^T$, as in Section 3.1.1 and for the reference state $\mathbf{s}_t^R = \begin{bmatrix}q_t^R\end{bmatrix}$, as in Section 3.1.2, the selection matrix $\mathbf{P}$ is $\begin{bmatrix}0 & 1\end{bmatrix}$. Since the reference state is one-dimensional, the symmetric weight matrix $\mathbf{Q}$ is 1. The reward function is differentiable with respect to the state, as required for the agent's update operations in the framework of DHP. The partial derivative is defined in Equation (3.10).

The motivation for the reward function originates from Linear Quadratic Regulator (LQR) problems in optimal control theory. It has been successfully applied RL approaches to reference tracking flight control, such as [27], [23] and [124]. Optionally, additional terms can be utilized to reward actions of small magnitude, in case the proposed reward function leads to an aggressive policy.

$$r(\mathbf{s}_t^R, \mathbf{s}_{t+1}) = -[\mathbf{P}\mathbf{s}_{t+1} - \mathbf{s}_t^R]^T \mathbf{Q} [\mathbf{P}\mathbf{s}_{t+1} - \mathbf{s}_t^R] \tag{3.9}$$

$$\frac{\partial r(\mathbf{s}_t^R, \mathbf{s}_{t+1})}{\partial \mathbf{s}_{t+1}} = -2[\mathbf{P}\mathbf{s}_{t+1} - \mathbf{s}_t^R]^T \mathbf{Q}\mathbf{P} \tag{3.10}$$

3.2. MDDHP Agent

In this section, a simplified version of the DHP agent, called MDDHP, is derived. Recall that, with the exemption of Action Dependent Heuristic Dynamic Programming (ADHDP), all frameworks within ACDs are model-based. Hence, they rely on a (known) model of the environment. To emphasize the use of the exact plant dynamics, the agent is referred to as MDDHP, instead of DHP. This section starts with the design of the memory structures utilized for the MDDHP agent. Subsequently, the update rules for the memory structures, in the context of the control task at hand, are derived. Last but not least, the agent's training strategy is presented.

3.2.1. Memory Structures

Recall that the DHP framework is an actor-critic approach, where the actor and the critic are memory structures representing the policy and the state-sensitivity of the state-value function, respectively. First, the network topology is described, followed by a discussion on activation functions. Subsequently, the choice of the optimizer is motivated.

Artificial Neural Network Topology

The actor and the critic are each represented by a feed-forward, Fully Connected Neural Network (FCNN), as depicted in Figure 3.1. Other topologies such as recurrent, nested or structured, when correctly targeted to the problem at hand, can provide superior performance compared to standard FCNNs. Whereas recurrent topologies such as Long Short Term Memory (LSTM) [31][8] are most favorable for time series data, nested topologies can take advantage of the physical properties of the environment [124]. Recently, Apple Inc. researchers have proposed a new topology for Deep Reinforcement Learning (DRL) control tasks, called Structured Control Net (SCN) [97]. It utilizes individual learning modules for linear and nonlinear control. For the preliminary analysis, the fundamental feed-forward, fully-connected topology is desired because of its simplicity.

Commonly, a time-consuming hyper-parameter search is conducted to find an optimal network structure. For the preliminary analysis however, these were derived from experiments conducted in

[107], [121] and [124]. A network structure with one hidden layer, containing six neurons, was selected for both the actor and critic, as illustrated in Figure 3.1.

For the input layer of the actor and critic, two options are considered. The state $\mathbf{s}$ concatenated with, the reference state $\mathbf{s}^R$ or with the reference error $\mathbf{Ps} - \mathbf{s}^R$. Hence, $\{\alpha, q, q^R\}$ or $\{\alpha, q, q - q^R\}$. The latter could be advantageous for the learning process, as it already provides important information, about the task to be learned, to the network. On the other hand, the data would be badly conditioned, as the tracking error has a different order of magnitude than the states, which can be detrimental to the learning process. A choice is made for $\{\alpha, q, q - q^R\}$.

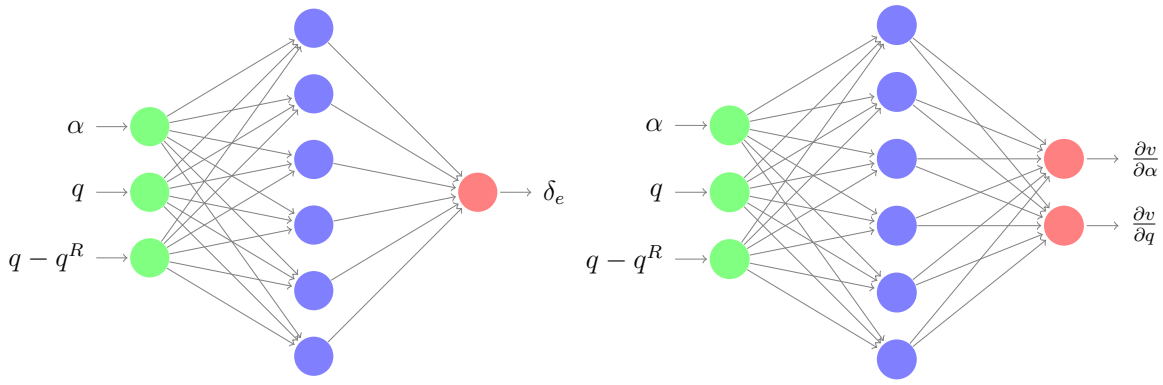

Figure 3.1: Artificial Neural Network topology of actor (left) and critic (right).

Activation Function

Activation functions introduce nonlinearity in ANNs. Hyperbolic tangent, sigmoid and Rectified Linear Unit (ReLU) are the most common activation functions. Compared to hyperbolic tangent and sigmoid activation, ReLUs generally accelerate the convergence of stochastic gradient descent, are less expensive to evaluate and allow a network to easily obtain sparse representations [40]. Furthermore, it does not have the undesirable property of saturation, which provokes vanishing gradients. However, for a shallow network, such as in this case, saturation and vanishing gradients are easily mitigated by careful initialization of the trainable parameters. ReLUs in networks with a few neurons can lead to a function approximation with insufficient smoothness. Unlike the hyperbolic tangent activation, sigmoid activation and ReLUs are not zero-centered and can, therefore, introduce undesirable dynamics in the gradient updates, especially for small batch sizes. Conclusively, hyperbolic tangent activation is utilized for the hidden layer. In order for the output to be unbounded, as necessary for a regression task, the output layer is linear.

Optimizer

In the following a comprehensive overview of optimization algorithms, as commonly used for deep learning tasks, is presented. Optimization algorithms that are unfeasible for high-dimensional data sets such as, second-order methods, are not considered. Gradient Descent (GD) is a fundamental optimization approach, where the parameters are updated by stepping in a weighted direction, opposite to the gradient of a loss function. This idea is demonstrated by Equation (3.11), with learning rate η, loss function $L(\cdot)$ and parameters $\mathbf{w}$.

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial L}{\partial \mathbf{w}} \tag{3.11}$$

Whereas standard GD computes one update based on the entire data-set, Stochastic Gradient Descent (SGD) does so for smaller individual batches of data. There are many challenges related to GD, as listed.

- Whereas a small learning rate leads to inadmissible slow convergence, a large learning rate can cause fluctuations around a minimum or even divergence.
- Especially for non-convex loss functions, training can get stuck in sub-optimal local minima, or saddle points [18].

- Learning rate scheduling methods, such as annealing, are defined in advance and are not able to adapt to the data's characteristics [17].
- Sparse data with features with various frequencies require parameter specific learning rates.

Research conducted to mitigate these issues has contributed to the development of many optimizers. In this section, only a comprehensive overview of optimizers is provided, as illustrated in Figure 3.2. A more extensive discussion is presented in [76].

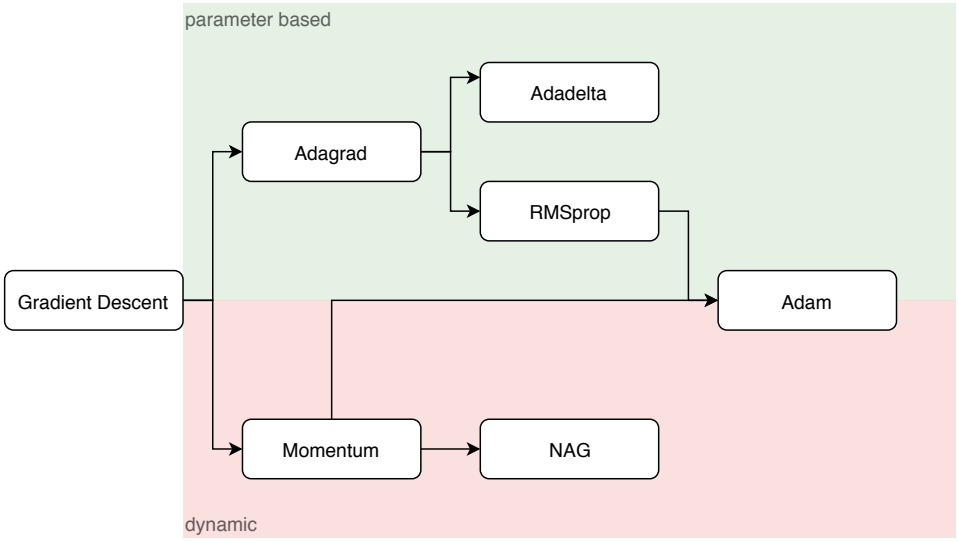

Figure 3.2: Overview of Gradient Descent optimization algorithms.

SGD suffers from oscillations and slow convergence in regions where the surface curves are steeper in one dimension than in another, so-called ravines. The ravines are common in the vicinity of local optima. Momentum [73] and Nesterov Accelerated Gradient (NAG) [61] use a recursive scheme to reduce oscillations and accelerate convergence, mitigating the first two challenges, as listed above. Adagrad [22] and its extensions Adadelta [120] and RMSprop update each parameter individually according to its importance. These parameter based methods eliminate the need for manual learning rate tuning and are able to deal with sparse data. Adam [38] combines the ideas of Momentum and RMSprop and can be regarded as the default optimizer for ANNs today.

The advantages of optimizers like Adam are most apparent for deep ANNs. As presented in Section 3.2.1, the ANNs utilized are rather shallow. Optimizers as RMSprop and Adam have shown to be numerically unstable when gradient get and stay close to zero. Generally, this is not an issue for deep learning problems, as commonly convergence is not achieved on that level, especially when regularization methods are applied. However, for the tracking task as formulated in Section 3.1, convergence with gradients approaching zero is inherent. Therefore, for the task at hand Adam performs worse than basic gradient descent. Furthermore, for the preliminary analysis, it is desirable to have a tractable learning process. Adaptive, parameter-based optimizers increase the complexity of the learning process. Last but not least, basic gradient descent has been successfully utilized in [121], [124] and [23]. Conclusively, standard SGD is utilized as optimization algorithm.

3.2.2. Update Rules

In this section, the derivation of the update rules for the actor and critic, in the context of the control task, as defined in Section 3.1, is presented. Figure 3.3 provides a schematic of the MDDHP framework.

Critic

Recall that in the framework of DHP the critic approximates the derivatives of the value function with respect to the state $\hat{\lambda}(\cdot)$, as presented by Equation (3.12), where $\mathbf{w}_t^C$ are the parameters of the critic at time t. The loss L_t^C for the critic, as defined by Equation (3.13), is derived according to the temporal difference error δ_t. In Equation (3.14) the aforementioned differentiability requirement for the reward function becomes apparent.

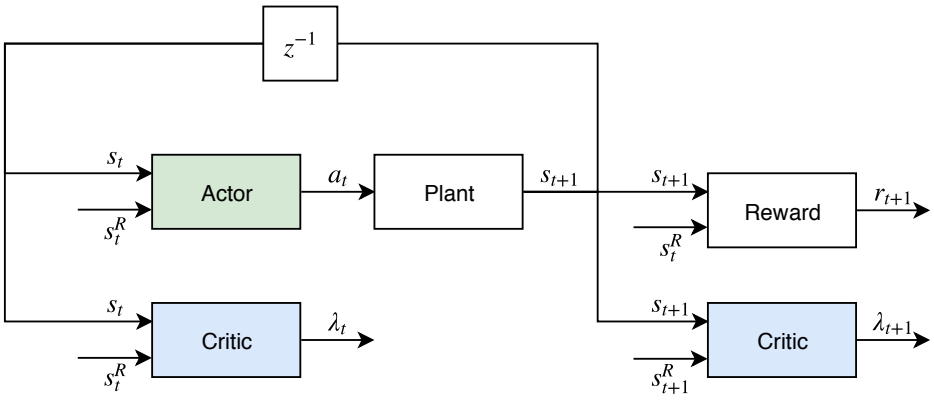

Figure 3.3: Schematic of MDDHP framework.

$$\hat{\lambda}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^C) \approx \lambda(\mathbf{s}_t, \mathbf{s}_t^R) = \frac{\partial v(\mathbf{s}_t, \mathbf{s}_t^R)}{\partial \mathbf{s}_t} \tag{3.12}$$

$$L_t^C = \frac{1}{2}\mathbf{e}_t \mathbf{e}_t^T \tag{3.13}$$

$$\begin{aligned} \mathbf{e}_t &= -\frac{\partial \delta_t}{\partial \mathbf{s}_t} \\ &= -\frac{\partial \Big[r(\mathbf{s}_t^R, \mathbf{s}_{t+1}) + \gamma v(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R) - v(\mathbf{s}_t, \mathbf{s}_t^R) \Big]}{\partial \mathbf{s}_t} \\ &\approx -\left[\frac{\partial r(\mathbf{s}_t^R, \mathbf{s}_{t+1})}{\partial \mathbf{s}_{t+1}} + \gamma \hat{\lambda}(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R, \mathbf{w}_t^C) \right] \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} + \hat{\lambda}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^C) \end{aligned} \tag{3.14}$$

The term $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t}$ in Equation (3.14) represents a partial derivative of the state-transition function. From Equation (3.15) it can be observed that the back-propagation process of the critic passes through the plant $f(\cdot)$ and the actor $\hat{\pi}(\cdot)$, where $\mathbf{w}_t^A$ are the parameters of the actor. Recall that the MDDHP framework makes use of the exact formulation of the partial derivatives of the state-transition function, as previously defined by Equation (3.3) and Equation (3.4).

$$\begin{aligned} \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} &= \frac{\partial f(\mathbf{s}_t, \mathbf{a}_t)}{\partial \mathbf{s}_t} + \frac{\partial f(\mathbf{s}_t, \mathbf{a}_t)}{\partial \mathbf{a}_t} \frac{\partial \mathbf{a}_t}{\partial \mathbf{s}_t} \\ &= \mathbf{F} + \mathbf{G} \frac{\partial \hat{\pi}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^A)}{\partial \mathbf{s}_t} \end{aligned} \tag{3.15}$$

Conclusively, the gradient of the loss function with respect to the trainable parameters $\mathbf{w}_t^C$ is defined by Equation (3.16). Although the term $\hat{\lambda}(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R, \mathbf{w}_t^C)$ is a function of $\mathbf{w}_t^C$, it is treated a constant in the partial derivative $\frac{\partial \mathbf{e}_t}{\partial \mathbf{w}_t^C}$, as it belongs to the prediction target $\frac{\partial r(\mathbf{s}_t^R, \mathbf{s}_{t+1})}{\partial \mathbf{s}_{t+1}} + \gamma \hat{\lambda}(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R, \mathbf{w}_t^C)$. Recent DRL frameworks, such as Deep Deterministic Policy Gradients (DDPG) [47] and Deep Q-Network (DQN) [56], make use of dedicated target networks to generate the targets for the temporal difference error computation, as elaborated in Section 2.4. That approach regularizes learning and increases stability, which is vital since these frameworks suffer from the so-called deadly triad [100]. MDDHP does not suffer from the deadly triad as, contrary to DDPG and DQN, it is an on-policy framework. The critic is updated by stepping in opposite direction to the gradient of the loss, as defined by Equation (3.17). Figure 3.4 provides a schematic of the update rule for the critic. The red lines illustrate the dependencies of the loss gradient. The use of a node's derivative instead of their output is implied by it being marked red. The node belonging to the memory structure being updated is marked with dotted lines.

$$\frac{\partial L_t^C}{\partial \mathbf{w}_t^C} = \frac{\partial L_t^C}{\partial \hat{\lambda}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^C)} \frac{\partial \hat{\lambda}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^C)}{\partial \mathbf{w}_t^C} = \mathbf{e}_t \frac{\partial \hat{\lambda}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^C)}{\partial \mathbf{w}_t^C} \tag{3.16}$$

$$\mathbf{w}_{t+1}^C = \mathbf{w}_t^C - \eta_t^C \frac{\partial L_t^C}{\partial \mathbf{w}_t^C} \tag{3.17}$$

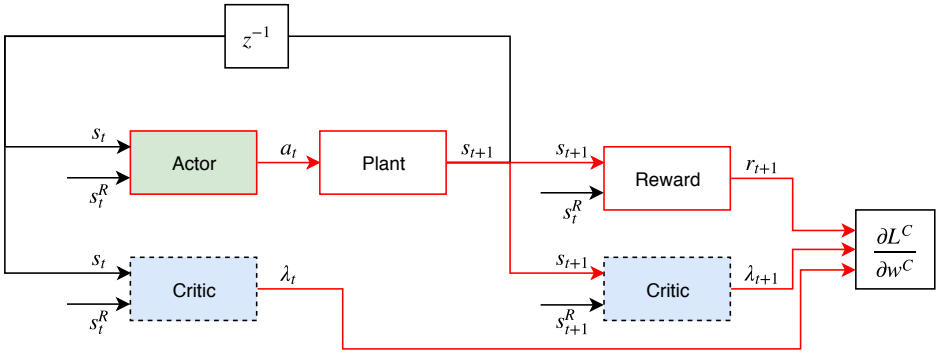

Figure 3.4: Back-propagation schematic for the critic update in the framework of MDDHP.

Actor

The goal is to update the actor towards a more optimal policy, which maximizes value function $v(\mathbf{s}_t, \mathbf{s}_t^R)$. Therefore, the loss of the actor L_t^A can be defined as in Equation (3.18). Equation (3.19) defines the gradient of the loss function with respect to the trainable parameters $\mathbf{w}_t^A$. Here, the back-propagation process passes through the critic $\hat{\lambda}(\cdot)$ and the plant $f(\cdot)$. For the term $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t}$, the exact formulation, as defined by Equation (3.4), is utilized. Conclusively, the update of the actor is conducted by stepping in opposite direction to the gradient of the loss, as defined by Equation (3.20). Figure 3.5 provides a schematic of the update rule for the actor.

$$L_t^A = -v(\mathbf{s}_t, \mathbf{s}_t^R) = -\Big[r(\mathbf{s}_t^R, \mathbf{s}_{t+1}) + \gamma v(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R)\Big] \tag{3.18}$$

$$\begin{aligned}
\frac{\partial L_t^A}{\partial \mathbf{w}_t^A} &= -\frac{\partial \Big[r(\mathbf{s}_t^R, \mathbf{s}_{t+1}) + \gamma v(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R)\Big]}{\partial \mathbf{w}_t^A} \\
&= -\left[\frac{\partial r(\mathbf{s}_t^R, \mathbf{s}_{t+1})}{\partial \mathbf{s}_{t+1}} + \gamma \frac{\partial v(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R)}{\partial \mathbf{s}_{t+1}}\right] \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t} \frac{\partial \mathbf{a}_t}{\partial \mathbf{w}_t^A} \\
&= -\left[\frac{\partial r(\mathbf{s}_t^R, \mathbf{s}_{t+1})}{\partial \mathbf{s}_{t+1}} + \gamma \hat{\lambda}(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R, \mathbf{w}_{t+1}^C)\right] \frac{\partial f(\mathbf{s}_t, \mathbf{a}_t)}{\partial \mathbf{a}_t} \frac{\partial \hat{\pi}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^A)}{\partial \mathbf{w}_t^A} \\
&= -\left[\frac{\partial r(\mathbf{s}_t^R, \mathbf{s}_{t+1})}{\partial \mathbf{s}_{t+1}} + \gamma \hat{\lambda}(\mathbf{s}_{t+1}, \mathbf{s}_{t+1}^R, \mathbf{w}_{t+1}^C)\right] \mathbf{G} \frac{\partial \hat{\pi}(\mathbf{s}_t, \mathbf{s}_t^R, \mathbf{w}_t^A)}{\partial \mathbf{w}_t^A}
\end{aligned} \tag{3.19}$$

$$\mathbf{w}_{t+1}^A = \mathbf{w}_t^A - \eta_t^A \frac{\partial L_t^A}{\partial \mathbf{w}_t^A} \tag{3.20}$$

3.2.3. Training Strategy

The update rules, as established in Section 3.2.2, are the mathematical basis necessary for the training strategy. In this section, the training strategy for the MDDHP agent is presented by means of a pseudo-code as in Algorithm 3.1.

Algorithm 3.1 starts by requiring the environment and agent parameters. During operation in a real aircraft, the simulation duration would be indefinite. In this example, the discount factor and learning

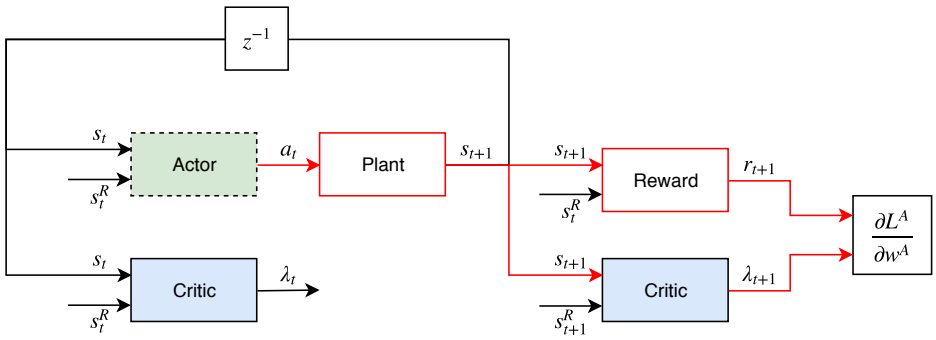

Figure 3.5: Back-propagation schematic for the actor update in the framework of MDDHP.

rates are not scheduled, but constant. Subsequently, the memory structures for the actor and critic are defined. Any kind of parameterization can be used as long as it is differentiable. In this chapter, ANN are used as elaborated in Section 3.2.1. Last but not least, the partial derivatives of the state-transition function and reward function are required.

Subsequent to the initialization of the memory structure parameters ($\mathbf{w}_0^A, \mathbf{w}_0^C$) and the initial state $\mathbf{s}_0$, the learning routine is commenced, as defined through lines 1 to 16. Algorithm 3.1 is divided into two parts. The first part consists of lines 14 and 15. They define the interaction of the agent with its environment. An action for the current state is predicted by the actor, which is carried out in the environment, resulting in the observation of the next state. Contrary to the second part, the first part is carried out for each simulation step. The second part defines the update procedure in lines 2 to 12. The update procedure is not carried out during the first step, as it requires the observations made through an interaction with the environment. An update operation is conducted after each interaction, based on that single observation. The second part starts by requesting the current reference state. Subsequently, the partial derivatives for the state-transition and reward function are evaluated in lines 4 to 6. Next, two predictions of the partial derivative of the value function are computed through the critic, as in lines 7 and 8. Last but not least, the change in parameters is computed and the parameters are updated for the critic and actor in lines 9 to 12. The parameters steps are computed based on the loss gradients as previously defined by Equation (3.16) and Equation (3.19).

The training strategy is motivated by two aspects, model dependency, and execution speed. A different approach is to conduct the updates based on state and reward predictions instead of observations [86]. There the goal is to initially compute an optimal action and subsequently update the actor toward that policy. Whereas that approach allows for an optimization of the agent before its interaction with the environment, it has some disadvantages and limitations. First, the state-transition and reward functions are required, whereas MDDHP only requires their partial derivatives and therefore has a lower model-dependency. Second, the number of operations increases substantially, as additional model and actor predictions are required, increasing the execution time. It is possible to conduct the critic update utilizing the updated actor parameters, or vice versa but requires an additional computation of the transition gradient in line 5 or critic prediction in line 8.

3.3. IDHP Agent

In this section the MDDHP agent as established in Section 3.2 is extended to the IDHP framework. The extension consists of the integration of the incremental model estimation, by means of Recursive Least Squares (RLS), to the MDDHP agent. This section starts by defining the incremental model and the RLS estimator, followed by its integration in the training strategy.

3.3.1. Incremental Model Estimation with RLS

Incremental models are commonly used to address nonlinearity in systems and rely on the assumption of sufficiently high sample frequency and slow-varying system dynamics [87] [90] [2]. The derivation of the incremental model involves linearization of the plant through a first-order Taylor series expansion, as presented in [122]. Equation (3.21) defines the linear, time-varying, incremental model.

Algorithm 3.1 Model Dependent Dual Heuristic Programming

Require:

environment parameters $\Delta t, T$

agent parameters γ, η^A, η^C

differentiable deterministic policy parameterization $\hat{\pi}(\mathbf{s}, \mathbf{s}^R, \mathbf{w}^A)$

differentiable state-value-derivative function parameterization $\hat{\lambda}(\mathbf{s}, \mathbf{s}^R, \mathbf{w}^C)$

state-transition function derivatives $\frac{\partial f(\mathbf{s},\mathbf{a})}{\partial \mathbf{s}}$, $\frac{\partial f(\mathbf{s},\mathbf{a})}{\partial \mathbf{a}}$

reward function derivative $\frac{\partial r(\mathbf{s}^R,\mathbf{s})}{\partial \mathbf{s}}$

Initialize:

$\mathbf{w}_0^A, \mathbf{w}_0^C, \mathbf{s}_0$

Compute:

for $i = 0$ **to** $int\left(\frac{T}{\Delta t}\right) - 1$ **do**

  get $\mathbf{s}_i^R$

  if $i \neq 0$ **then**

    $\frac{\partial \mathbf{s}_i}{\partial \mathbf{a}_{i-1}} \leftarrow \frac{\partial f(\mathbf{s}_{i-1},\mathbf{a}_{i-1})}{\partial \mathbf{a}_{i-1}}$

    $\frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} \leftarrow \frac{\partial f(\mathbf{s}_{i-1},\mathbf{a}_{i-1})}{\partial \mathbf{s}_{i-1}} + \frac{\partial \mathbf{s}_i}{\partial \mathbf{a}_{i-1}} \frac{\partial \hat{\pi}(\mathbf{s}_{i-1},\mathbf{s}_{i-1}^R,\mathbf{w}_{i-1}^A)}{\partial \mathbf{s}_{i-1}}$

    $\frac{\partial r_i}{\partial \mathbf{s}_i} \leftarrow \frac{\partial r(\mathbf{s}_{i-1}^R,\mathbf{s}_i)}{\partial \mathbf{s}_i}$

    $\boldsymbol{\lambda}_{i-1} \leftarrow \hat{\lambda}(\mathbf{s}_{i-1}, \mathbf{s}_{i-1}^R, \mathbf{w}_{i-1}^C)$

    $\boldsymbol{\lambda}_i \leftarrow \hat{\lambda}(\mathbf{s}_i, \mathbf{s}_i^R, \mathbf{w}_{i-1}^C)$

    $\Delta \mathbf{w}_i^A \leftarrow \eta^A \left[\frac{\partial r_i}{\partial \mathbf{s}_i} + \gamma \boldsymbol{\lambda}_i\right] \frac{\partial \mathbf{s}_i}{\partial \mathbf{a}_{i-1}} \frac{\partial \hat{\pi}(\mathbf{s}_{i-1},\mathbf{s}_{i-1}^R,\mathbf{w}_{i-1}^A)}{\partial \mathbf{w}_{i-1}^A}$

    $\Delta \mathbf{w}_i^C \leftarrow -\eta^C \left[-\left[\frac{\partial r_i}{\partial \mathbf{s}_i} + \gamma \boldsymbol{\lambda}_i\right] \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} + \boldsymbol{\lambda}_{i-1}\right] \frac{\partial \boldsymbol{\lambda}_{i-1}}{\partial \mathbf{w}_{i-1}^C}$

    $\mathbf{w}_i^A \leftarrow \mathbf{w}_{i-1}^A + \Delta \mathbf{w}_i^A$

    $\mathbf{w}_i^C \leftarrow \mathbf{w}_{i-1}^C + \Delta \mathbf{w}_i^C$

  end if

  $\mathbf{a}_i \leftarrow \hat{\pi}(\mathbf{s}_i, \mathbf{s}_i^R, \mathbf{w}_i^A)$

  get $\mathbf{s}_{i+1}$ by taking action $\mathbf{a}_i$

end for

$$\Delta \mathbf{s}_{t+1} = \mathbf{F}_{t-1}\Delta \mathbf{s}_t + \mathbf{G}_{t-1}\Delta \mathbf{a}_t \tag{3.21}$$

The time varying state and input matrices $\mathbf{F}$ and $\mathbf{G}$ are identified online with a RLS estimator. The estimates of these matrices $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$ are defined in the parameter matrix $\hat{\mathbf{\Theta}}$, as presented in Equation (3.22). The parameter matrix is accompanied by a covariance matrix $\mathbf{\Lambda}$, which expresses a measure of confidence of the parameter estimates.

$$\hat{\mathbf{\Theta}}_{t-1} = \begin{bmatrix} \hat{\mathbf{F}}_{t-1}^T \\ \hat{\mathbf{G}}_{t-1}^T \end{bmatrix} \tag{3.22}$$

The update process of the RLS estimator starts with a prediction of state increments $\Delta\hat{\mathbf{s}}$ based on the latest measurements $\mathbf{X}$ and the current parameter estimates $\hat{\mathbf{\Theta}}$, as defined in Equation (3.23). The measurement matrix $\mathbf{X}$ is defined in Equation (3.24) and consists of the previous state and action increments. Next, the prediction error $\boldsymbol{\epsilon}$, named innovation, is computed, as defined in Equation (3.25). Last but not least, the parameter estimates and covariance matrix are updated based on Equation (3.26) and Equation (3.27). $\kappa \in [0, 1]$ is the forgetting factor, which exponentially weights older measurements.

$$\Delta\hat{\mathbf{s}}_{t+1}^T = \mathbf{X}_t^T \hat{\mathbf{\Theta}}_{t-1} \tag{3.23}$$

$$\mathbf{X}_t = \begin{bmatrix} \Delta \mathbf{s}_t \\ \Delta \mathbf{a}_t \end{bmatrix} \tag{3.24}$$

$$\boldsymbol{\epsilon}_t = \Delta \mathbf{s}_{t+1}^T - \Delta\hat{\mathbf{s}}_{t+1}^T \tag{3.25}$$

$$\hat{\mathbf{\Theta}}_t = \hat{\mathbf{\Theta}}_{t-1} + \frac{\mathbf{\Lambda}_{t-1}\mathbf{X}_t}{\kappa + \mathbf{X}_t^T \mathbf{\Lambda}_{t-1}\mathbf{X}_t}\boldsymbol{\epsilon}_t \tag{3.26}$$

$$\mathbf{\Lambda}_t = \frac{1}{\kappa}\left[\mathbf{\Lambda}_{t-1} - \frac{\mathbf{\Lambda}_{t-1}\mathbf{X}_t\mathbf{X}_t^T\mathbf{\Lambda}_{t-1}}{\kappa + \mathbf{X}_t^T\mathbf{\Lambda}_{t-1}\mathbf{X}_t}\right] \tag{3.27}$$

Different estimators can be utilized to estimate the state and input matrices of the incremental model. The choice for RLS is motivated by its advantage over e.g. piece-wise Ordinary Least Squares (OLS). OLS utilizes matrix inversion, which can fail due to insufficient system excitation.

3.3.2. Training Strategy

The incremental model with RLS estimation is added to the previous framework of MDDHP to form the IDHP framework. Its training strategy is presented by the pseudo code in Algorithm 3.2. In this section the differences between Algorithm 3.1 and Algorithm 3.2 are presented.

Due to the use of the incremental model as an estimate for the state-transition derivatives, the IDHP framework does not require knowledge about the plant's dynamics. The forgetting factor of the RLS estimator is added as a required agent parameter. Additionally to the initialization of the memory structures and state, the parameter estimate and covariance matrix are initialized. Similarly to Algorithm 3.1, Algorithm 3.2 consists of an update part in lines 3 to 25 and an interaction part in lines 26 and 27. In contrary to Algorithm 3.1, Algorithm 3.2 starts updating at the second time step, as the incremental model requires the observation of increments. The update of the memory structures in lines 8 to 16 is equivalent to Algorithm 3.1 with the exception of the state-transition derivatives. Subsequently to the update of the memory structures, the parameter estimate and covariance matrix are updated in lines 17 to 24, as elaborated in Section 3.3.2.

3.4. Results and Discussion

In this section the testing of the agents, as derived in Section 3.2 and Section 3.3, is divided in two parts. The first part, named fundamental testing, starts by assessing the ability of the agent to learn the reference tracking task at hand for the simplified PH-LAB model, given perfect knowledge about the plants state-transition derivatives. This is achieved by investigating the performance of the MDDHP agent. Subsequently, the knowledge about the plant is removed and an incremental model with RLS identification is introduced to the agent. Through the analysis of the IDHP agent, the model identification

Algorithm 3.2 Incremental Dual Heuristic Programming

Require:

environment parameters $\Delta t, T$

agent parameters $\gamma, \eta^A, \eta^C, \kappa$

differentiable deterministic policy parameterization $\hat{\pi}(\mathbf{s}, \mathbf{s}^R, \mathbf{w}^A)$

differentiable state-value-derivative function parameterization $\hat{\lambda}(\mathbf{s}, \mathbf{s}^R, \mathbf{w}^C)$

reward function derivative $\frac{\partial r(\mathbf{s}^R, \mathbf{s})}{\partial \mathbf{s}}$

Initialize:

$\mathbf{w}_0^A, \mathbf{w}_0^C, \hat{\mathbf{\Theta}}_0, \mathbf{\Lambda}_0, \mathbf{s}_0$

Compute:

for $i = 0$ **to** $int\left(\frac{T}{\Delta t}\right) - 1$ **do**
  get $\mathbf{s}_i^R$
  if $i = 1$ **then**
    $\mathbf{w}_i^A \leftarrow \mathbf{w}_{i-1}^A$
    $\mathbf{w}_i^C \leftarrow \mathbf{w}_{i-1}^C$
  end if
  if $i > 1$ **then**
    $\begin{bmatrix} \hat{\mathbf{F}}_{i-2}^T \\ \hat{\mathbf{G}}_{i-2}^T \end{bmatrix} \leftarrow \hat{\mathbf{\Theta}}_{i-2}$
    $\frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} \leftarrow \hat{\mathbf{F}}_{i-2} + \hat{\mathbf{G}}_{i-2} \frac{\partial \hat{\pi}(\mathbf{s}_{i-1}, \mathbf{s}_{i-1}^R, \mathbf{w}_{i-1}^A)}{\partial \mathbf{s}_{i-1}}$
    $\frac{\partial r_i}{\partial \mathbf{s}_i} \leftarrow \frac{\partial r(\mathbf{s}_{i-1}^R, \mathbf{s}_i)}{\partial \mathbf{s}_i}$
    $\boldsymbol{\lambda}_{i-1} \leftarrow \hat{\lambda}(\mathbf{s}_{i-1}, \mathbf{s}_{i-1}^R, \mathbf{w}_{i-1}^C)$
    $\boldsymbol{\lambda}_i \leftarrow \hat{\lambda}(\mathbf{s}_i, \mathbf{s}_i^R, \mathbf{w}_{i-1}^C)$
    $\Delta \mathbf{w}_i^A \leftarrow \eta^A \left[\frac{\partial r_i}{\partial \mathbf{s}_i} + \gamma \boldsymbol{\lambda}_i\right] \hat{\mathbf{G}}_{i-2} \frac{\partial \hat{\pi}(\mathbf{s}_{i-1}, \mathbf{s}_{i-1}^R, \mathbf{w}_{i-1}^A)}{\partial \mathbf{w}_{i-1}^A}$
    $\Delta \mathbf{w}_i^C \leftarrow -\eta^C \left[-\left[\frac{\partial r_i}{\partial \mathbf{s}_i} + \gamma \boldsymbol{\lambda}_i\right] \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} + \boldsymbol{\lambda}_{i-1}\right] \frac{\partial \boldsymbol{\lambda}_{i-1}}{\partial \mathbf{w}_{i-1}^C}$
    $\mathbf{w}_i^A \leftarrow \mathbf{w}_{i-1}^A + \Delta \mathbf{w}_i^A$
    $\mathbf{w}_i^C \leftarrow \mathbf{w}_{i-1}^C + \Delta \mathbf{w}_i^C$
    $\Delta \mathbf{s}_{i-1} \leftarrow \mathbf{s}_{i-1} - \mathbf{s}_{i-2}$
    $\Delta \mathbf{a}_{i-1} \leftarrow \mathbf{a}_{i-1} - \mathbf{a}_{i-2}$
    $\Delta \mathbf{s}_i \leftarrow \mathbf{s}_i - \mathbf{s}_{i-1}$
    $\mathbf{X}_{i-1} \leftarrow \begin{bmatrix} \Delta \mathbf{s}_{i-1} \\ \Delta \mathbf{a}_{i-1} \end{bmatrix}$
    $\Delta \hat{\mathbf{s}}_i^T \leftarrow \mathbf{X}_{i-1}^T \hat{\mathbf{\Theta}}_{i-2}$
    $\boldsymbol{\epsilon}_{i-1} \leftarrow \Delta \mathbf{s}_i^T - \Delta \hat{\mathbf{s}}_i^T$
    $\hat{\mathbf{\Theta}}_{i-1} \leftarrow \hat{\mathbf{\Theta}}_{i-2} + \frac{\mathbf{\Lambda}_{i-2} \mathbf{X}_{i-1}}{\kappa + \mathbf{X}_{i-1}^T \mathbf{\Lambda}_{i-2} \mathbf{X}_{i-1}} \boldsymbol{\epsilon}_{i-1}$
    $\mathbf{\Lambda}_{i-1} \leftarrow \frac{1}{\kappa} \left[\mathbf{\Lambda}_{i-2} - \frac{\mathbf{\Lambda}_{i-2} \mathbf{X}_{i-1} \mathbf{X}_{i-1}^T \mathbf{\Lambda}_{i-2}}{\kappa + \mathbf{X}_{i-1}^T \mathbf{\Lambda}_{i-2} \mathbf{X}_{i-1}}\right]$
  end if
  $\mathbf{a}_i \leftarrow \hat{\pi}(\mathbf{s}_i, \mathbf{s}_i^R, \mathbf{w}_i^A)$
  get $\mathbf{s}_{i+1}$ by taking action $\mathbf{a}_i$
end for

performance and the capability of the agent to learn the task with estimated state-transition gradients is assessed. After the analysis of the agent's capability to learn the control task for the simplified PH-LAB model, the second part, name advanced testing, analyzes the capability of the IDHP agent to deal with more complex scenarios. These complex scenarios evaluate the agent's robustness to untrimmed initialization and measurement noise, and its adaptability to changes in the plant.

3.4.1. Fundamental Testing

In this section, the agent's capability of successfully conducting the pitch rate reference task on the simplified PH-LAB model is tested. Initially with exact knowledge about the plant and subsequently by means of RLS identification of an incremental model.

MDDHP

In the following the MDDHP agent as established in Section 3.2 is applied to the short period plant model to conduct a pitch rate reference tracking control task, as defined in Section 3.1, according to Algorithm 3.1. The agent's parameters are determined empirically by experimentation and common guidelines proposed in [86]. The agent's parameters, memory structure initialization, and initial state are summarized in Table 3.3. The memory structure parameters, which consists of weights and no biases, are distributed according to a truncated Gaussian distribution with zero mean and a standard deviation of 0.1. The plant is initialized in a trimmed state with $\mathbf{s}_0 = \mathbf{0}$.

Table 3.3: Agent parameters, memory structure initialization and initial state.

Parameter	Value/Setting
γ, η^A, η^C	$0.8,\ 10,\ 20$
$\mathbf{w}_0^A, \mathbf{w}_0^C$	$\mathcal{N}_{trunc}(0,\ 0.1^2)$
$\mathbf{s}_0$	$\mathbf{0}$

Figure 3.6 illustrates the results for a simulation with a duration of 40 seconds. The first subplot displays the plant's state and the reference signal. The other subplots below illustrate the agent's action and received reward, respectively. From Figure 3.6 it can be observed that the agent is able to learn a policy, which allows successful tracking of the pitch rate. Both the states and action show smooth control behavior. The reward quickly converges to the vicinity of its optimal zero value. After approximately 2 seconds the agent starts to steer the aircraft towards the desired pitch rate. This behavior is coherent with the memory structure parameters as depicted in Figure 3.7. Both the critic and actor are close to convergence after 2 seconds. In contrary to the critic, the actor is still slightly improving until converging to a steady distribution after 7 seconds, which can also be observed in Figure 3.6 from the decrease in tracking lag. In Figure 3.7 both the initial truncated Gaussian distribution and the final steady distribution can be identified. It can be argued that initializing the memory structures with a larger standard deviation could increase the convergence speed of the agent.

IDHP

As for the MDDHP agent, simulation results are generated for the IDHP agent using the same parameters as listed in Table 3.3. Table 3.4 lists the additional parameters distinct to the IDHP agent. In contrary to the MDDHP agent, the IDHP agent relies on the online identification of the incremental model, as it does not possess any knowledge about the plant's dynamics. Figure 3.8 illustrates the simulation results and Figure 3.9 the memory structure parameters for the IDHP agent. Both figures show similar control behavior to the MDDHP agent, implying good control performance of the IDHP agent. Before proceeding with a comparison between both agents, the incremental model with RLS system identification of the IDHP agent is evaluated. Figure 3.10 illustrates the absolute error of the estimates of the state and input matrices. It can be observed that the estimation error drops to small values within the first few seconds, providing the agent with close to exact state-transition gradients.

Last but not least, the previously attained simulation results for the MDDHP and IDHP agents are unified, as illustrated by Figure 3.11. Figure 3.11 emphasizes the equivalent performance of IDHP to MDDHP. It can be observed that the IDHP has slightly more overshoot and requires slightly more time to converge than MDDHP, which can be attributed to the small estimation error of the state-transition gradients during the initial phase.

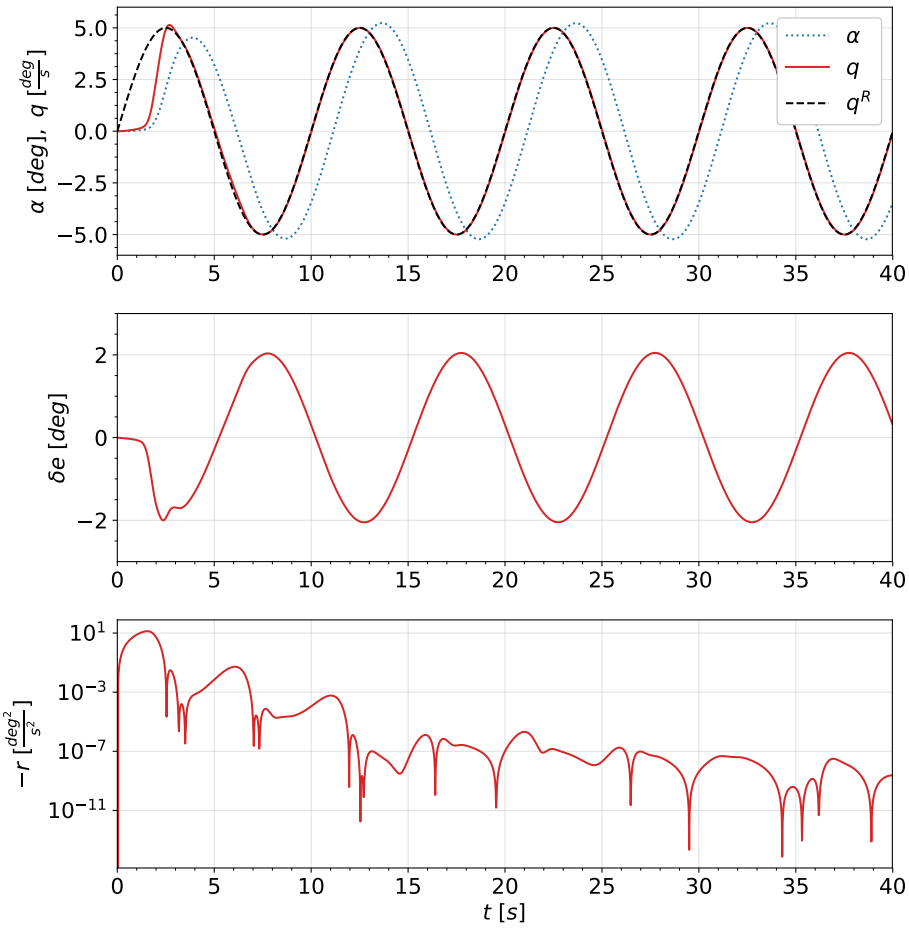

Figure 3.6: Pitch rate reference tracking on simplified PH-LAB model with MDDHP agent.

Table 3.4: Forgetting factor, initial parameter matrix and initial covariance matrix of IDHP agent.

Parameter	Value/Setting
κ	0.8
$\hat{\mathbf{\Theta}}_0$	$\mathbf{0}$
$\mathbf{\Lambda}_0$	$100\mathbf{I}$

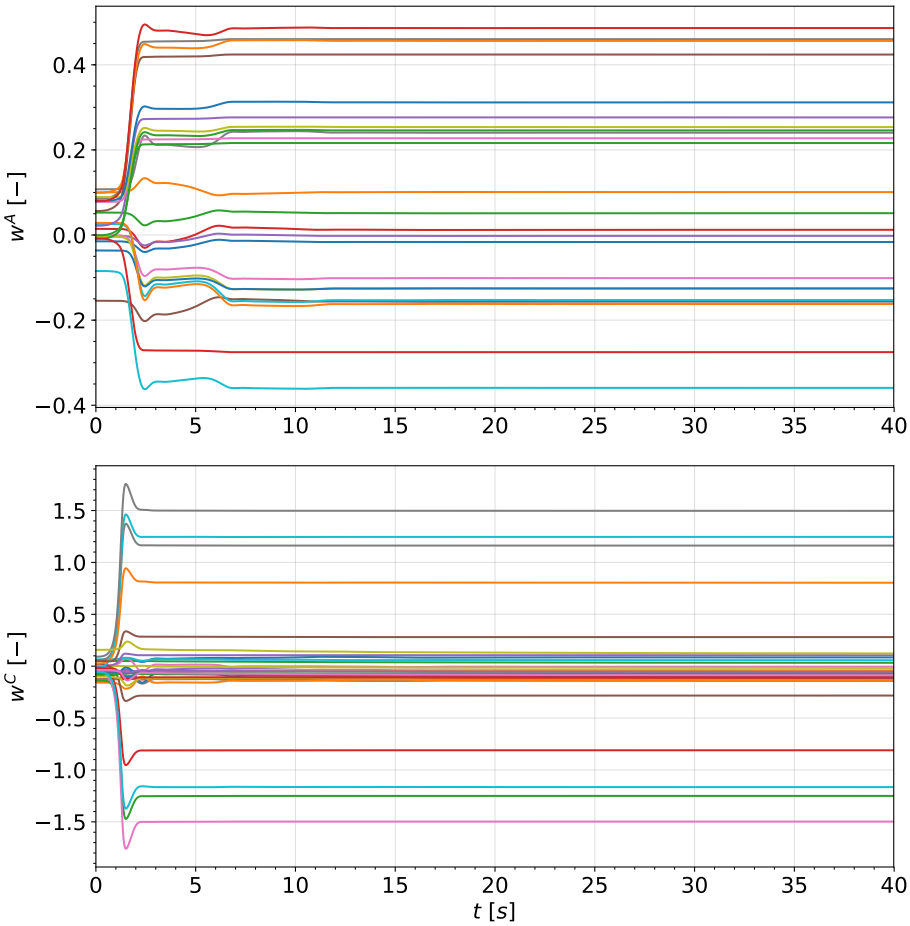

Figure 3.7: Memory structure parameters of actor and critic for MDDHP agent during pitch rate reference tracking on simplified PH-LAB model.

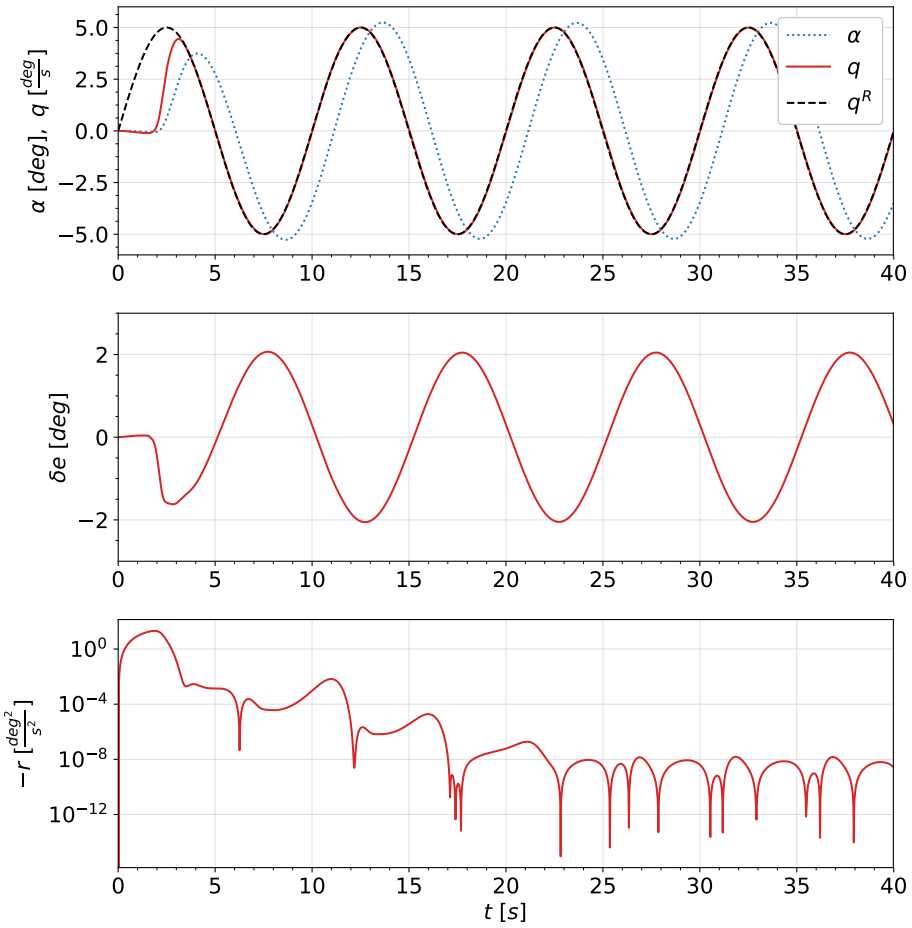

Figure 3.8: Pitch rate reference tracking on simplified PH-LAB model with IDHP agent.

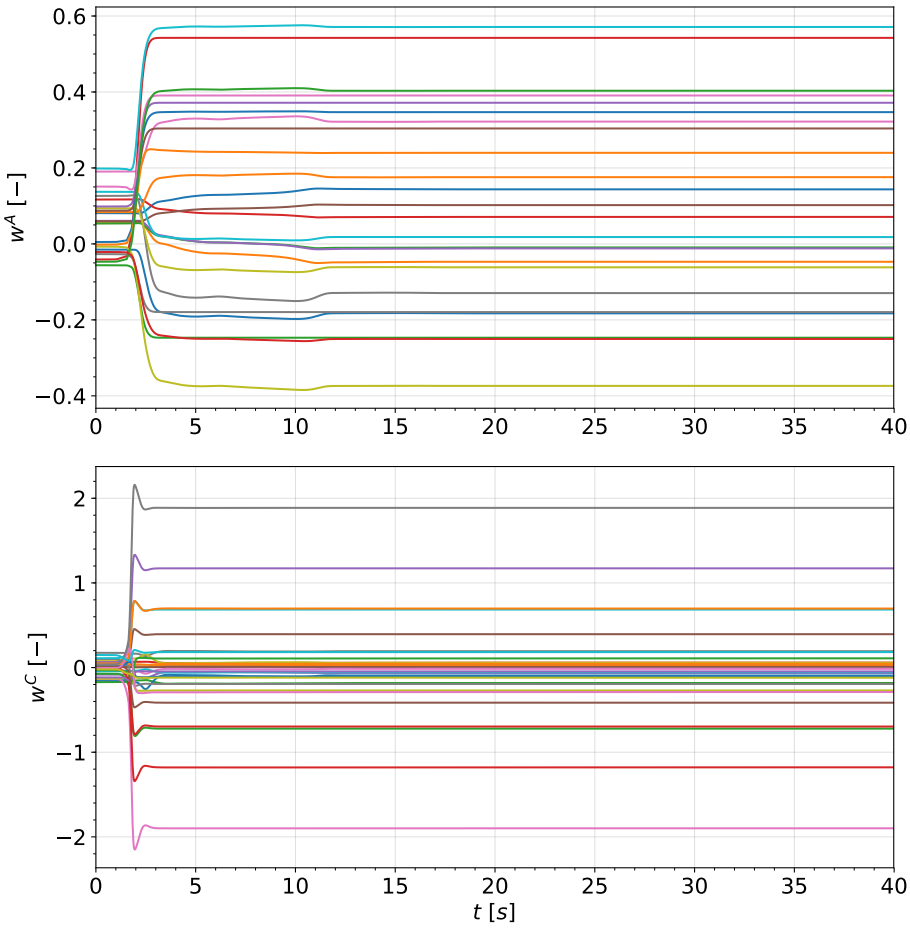

Figure 3.9: Memory structure parameters of actor and critic for IDHP agent during pitch rate reference tracking on simplified PH-LAB model.

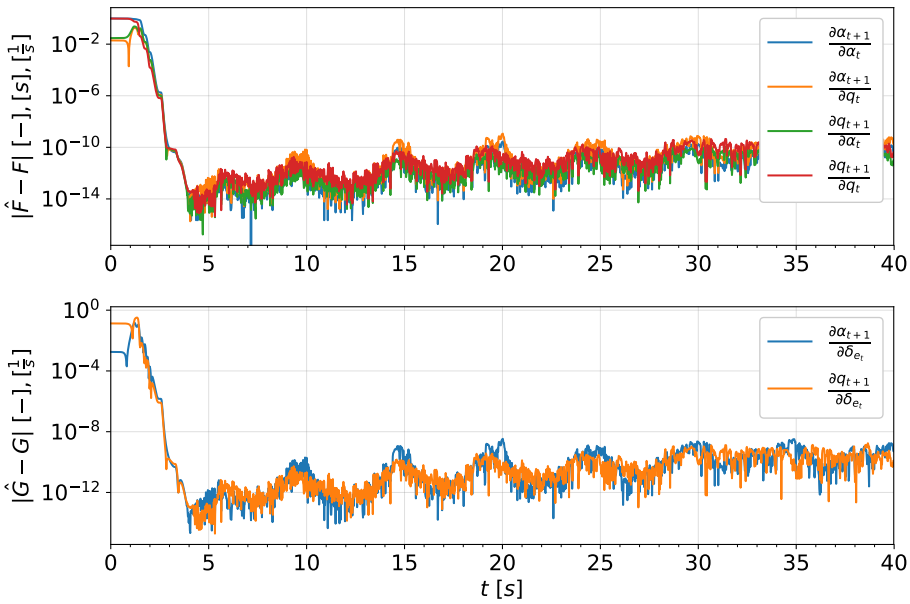

Figure 3.10: Absolute estimation error of state and input matrices of incremental model with RLS identification.

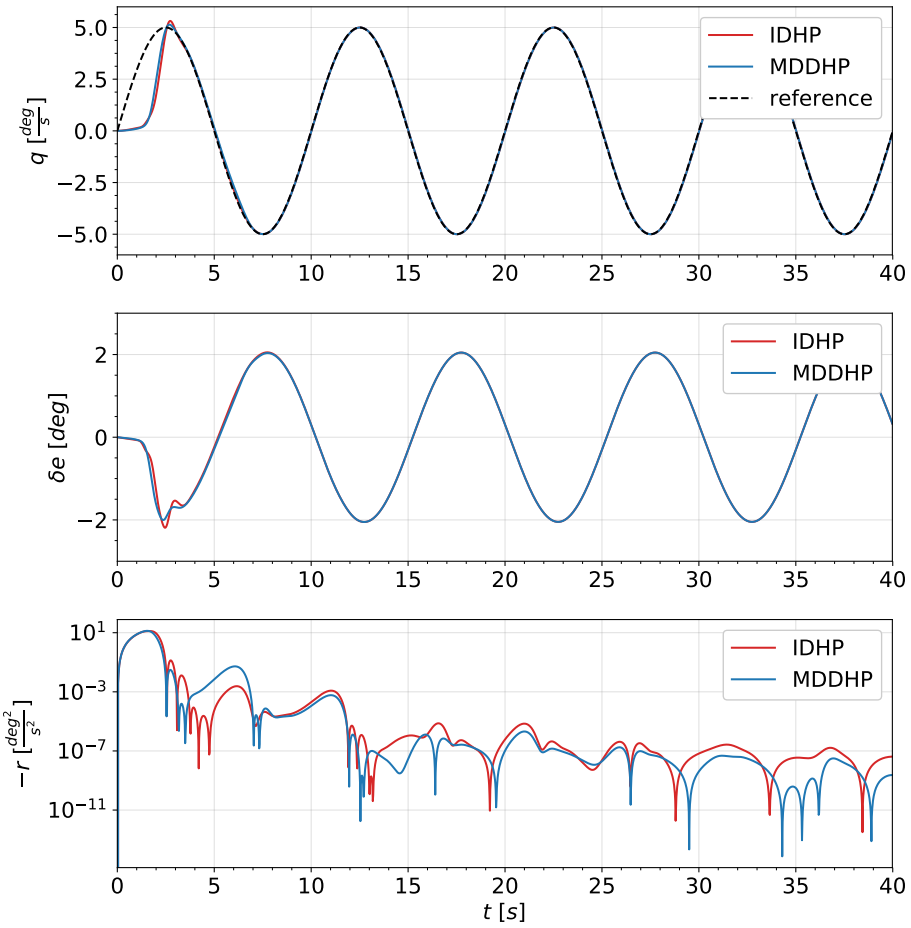

Figure 3.11: Pitch rate reference tracking on simplified PH-LAB model comparison between IDHP and MDDHP.

Synopsis
From the analysis of the MDDHP agent, it can be concluded that the framework as established in Section 3.2 is capable of learning a policy for conducting a pitch rate reference task for the simplified PH-LAB model, given perfect knowledge about the plant. When no knowledge about the plant is available, an incremental model with RLS identification can be successfully utilized, as can be concluded from the analysis of the IDHP agent.

3.4.2. Advanced Testing

Next, the robustness and adaptability of the IDHP agent are assessed. First, its performance is evaluated for untrimmed initialization and noisy state measurements. Subsequently, modifications to the plant are introduced during simulation.

Untrimmed Initialization
In the previous experiments, the plant was initialized at an initial trimmed state $\mathbf{s}_0 = \mathbf{0}$. However, initial operation at a trimmed state is unlikely, nor can the trimmed state be determined for an unknown plant. A good controller should be able to deal with a variety of untrimmed initial states, without compromising its ability to control the aircraft. A batch of 500 runs is simulated with random state initialization. α_0 and q_0 are sampled from uniform distributions with the the ranges $(-5, 5][deg]$ and $(-3, 3][\frac{deg}{s}]$, respectively. The agent's parameters are as listed by Table 3.3 and Table 3.4. Figure 3.12 illustrates the mean and the min-max bound of the ensemble of 500 runs. It can be observed that for every single run the agent is able to attain control of the aircraft. After seven seconds no effect of random initialization can be observed anymore in the pitch rate tracking performance.

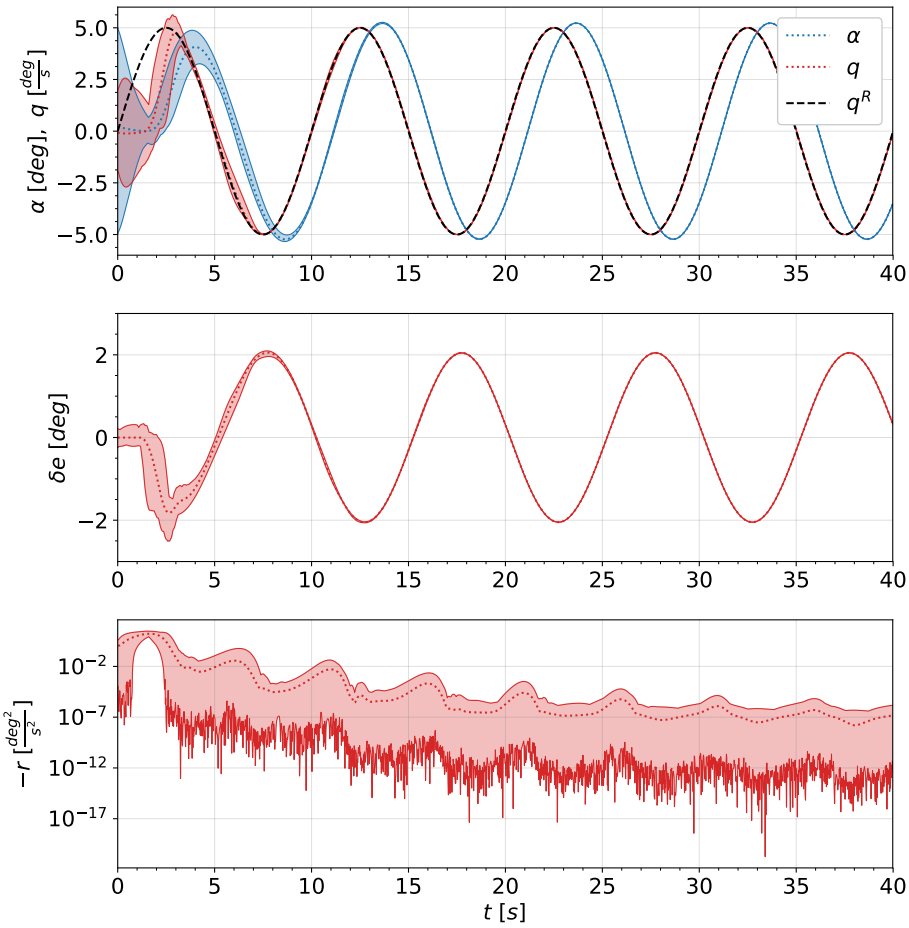

Figure 3.12: Mean and min-max bound of 500 runs of IDHP pitch rate reference tracking on simplified PH-LAB model with untrimmed, uniform, random state initialization.

Measurement Noise

Measurements of the plant's state acquired with sensors are polluted with noise. Therefore, it is important for the agent to be robust against a certain level of measurement noise. High-frequency zero-mean Gaussian noise is superimposed on the states α and q, with a standard deviation of $0.05\ [deg]$ and $0.005\ \left[\frac{deg}{s}\right]$, respectively. The order of intensity for the noise was derived from [124]. The agent is simulated with the parameters as listed in Table 3.3 and Table 3.4. From Figure 3.13 it can be observed that the agent is still capable of controlling the aircraft with noisy state measurements. The settling time, however, is longer as the polluted state measurements interfere with the identification of the incremental model, which difficultates the agent's learning process. The agent converges to a close to optimal policy, with a reward in the same order of magnitude as the Gaussian noise imposed on to the pitch rate.

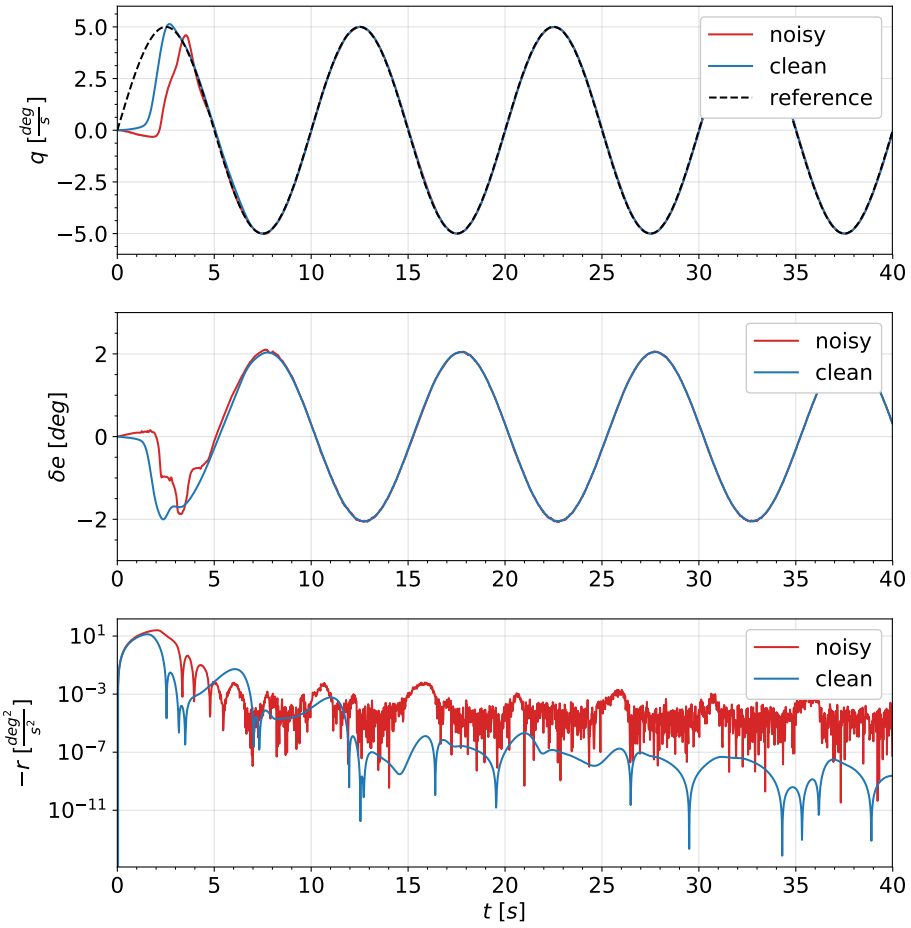

Figure 3.13: Pitch rate reference tracking on simplified PH-LAB model comparison between clean and noisy state measurements.

Fault-Tolerance

The branch of fault-tolerant control includes many aspects, such as fault detection and control adaption. In this section, the aim is to observe the capability of the agent to adapt to a sudden change in the plant's dynamics. For this experiment, a simple fault-detection is assumed. Fault detection is done through the observation of the innovation term of the RLS estimator, as derived in Equation (3.25). A fault is detected if the innovation term exceeds the empirically determined threshold of $0.0005\ [deg]$ and $0.001\ \left[\frac{deg}{s}\right]$ for α and q, respectively. At the occurrence of a detection, the agent's memory structure, as well as the RLS estimator, are reinitialized. The agent's parameters are as summarized by Table 3.3 and Table 3.4. A change is induced to the plant at $t = 20\ [s]$ by setting the input matrix to $\mathbf{G} = -\mathbf{G}$, requiring inverted operation of the elevator. From Figure 3.14 it can be observed that the agent is

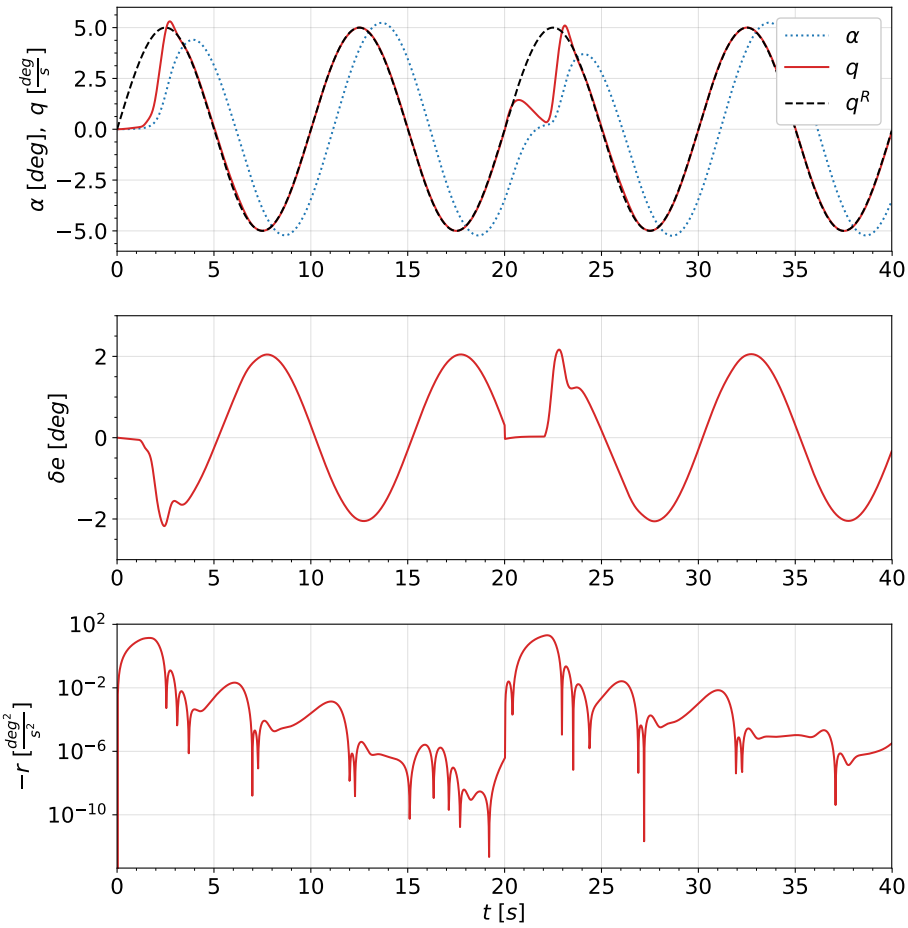

Figure 3.14: Pitch rate reference tracking on simplified PH-LAB model with IDHP agent in presence of sudden change to the plant's input matrix.

capable of dealing with the sudden change in the plant's dynamics. In fact, it behaves as if a new simulation started at $t = 20\ [s]$ at an untrimmed state and with different plant dynamics. The policy learns to invert the elevator control to successfully regain control of the aircraft.

Synopsis

The experiments conducted in the advanced testing phase aimed to asses the robustness and adaptability of the IDHP agent. When confronted with untrimmed initialization the agent was able to attain satisfactory control performance in all of the 500 conducted trials. Similarly, during an aggressive change on the control surface dynamics, the agent was able to quickly adapt without loss of control. Although the effects of measurement noise are clearly observable in the model estimation, the agent is still able to achieve satisfactory control performance. Conclusively, it can be remarked that the agent demonstrates a high level of adaptability and robustness when applied to the simplified model if the PH-LAB research aircraft.

3.5. Conclusion

Through the exact formulation of the problem in Section 3.1 and the derivation of the MDDHP and IDHP agents in Section 3.2 and Section 3.3 an extensive amount of implementation experience was gained. The results of the discussion of the experiments conducted in Section 3.4 demonstrated that the proposed baseline framework is both reproducible and applicable to the PH-LAB research aircraft. As a result, the preliminary analysis completes the research goal (G2) and answers the research question RQ1.

III

Additional Results and Discussions

4

Simulation of Online Operation in Different Flight Regimes

In the paper as presented in Part I the agent is initially trained during a dedicated online training phase. The online training phase is initiated at a trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Subsequently, the agent is evaluated on its performance to operate the aircraft for a representative maneuver which consists of a prolonged climb and a subsequent descent (with $5\ \frac{m}{s}$), combined with left and right turns (of $25\ deg$ and $20\ deg$). To demonstrate the agent's ability to operate the aircraft in different (not previously experienced) regimes, the maneuver is conducted for 4 flight regimes, as listed in Table 4.1. A set of 100 runs is simulated for each flight condition. As illustrated in Table 4.1, for all 100 runs in each flight condition, the agent has not failed.

Figures 4.1 to 4.4 depict the time histories of the minimum to maximum bound of all 100 runs for each flight condition. In all cases, it can be observed that the agent is able to fly the reference profile as commanded by the PID controllers. Furthermore, the conducted turns are well-coordinated as the sideslip is minimized, by the actuation of the rudder. In addition to the initial satisfactory control behavior, the agent keeps improving as it interacts with the environment. This is most noticeable from the narrowing of the min-max bound of the time history of the sideslip angle and rudder deflection. In Fig. 4.2 the narrowing of the min-max bound is only clearly observed for the time history of the rudder deflection and not the sideslip angle. This originates from the fact that the sideslip angle is already very small.

Similarly, the longitudinal learning controller succeeds in tracking the climb and descent profile for all flight conditions. As elaborated in Part I the learning controllers do not conceive a notion of the airspeed as a distinct state, but can only experience it as an external, temporary change in the perceived plant dynamics. Therefore, the agent's tracking performance temporarily degrades during quick changes in the airspeed conducted by the aircraft's internal airspeed controller.

Table 4.1: Description and sample failure rate of different flight conditions each simulated 100 times during the online operation phase. For all cases the agent is initialized at its online pretrained state conducted at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$.

Flight Condition ID	$H_0\ m$	$V_{tas_0}\ \frac{m}{s}$	Failure Rate
FC0	2000	90	0%
FC1	2000	140	0%
FC2	5000	90	0%
FC3	5000	140	0%

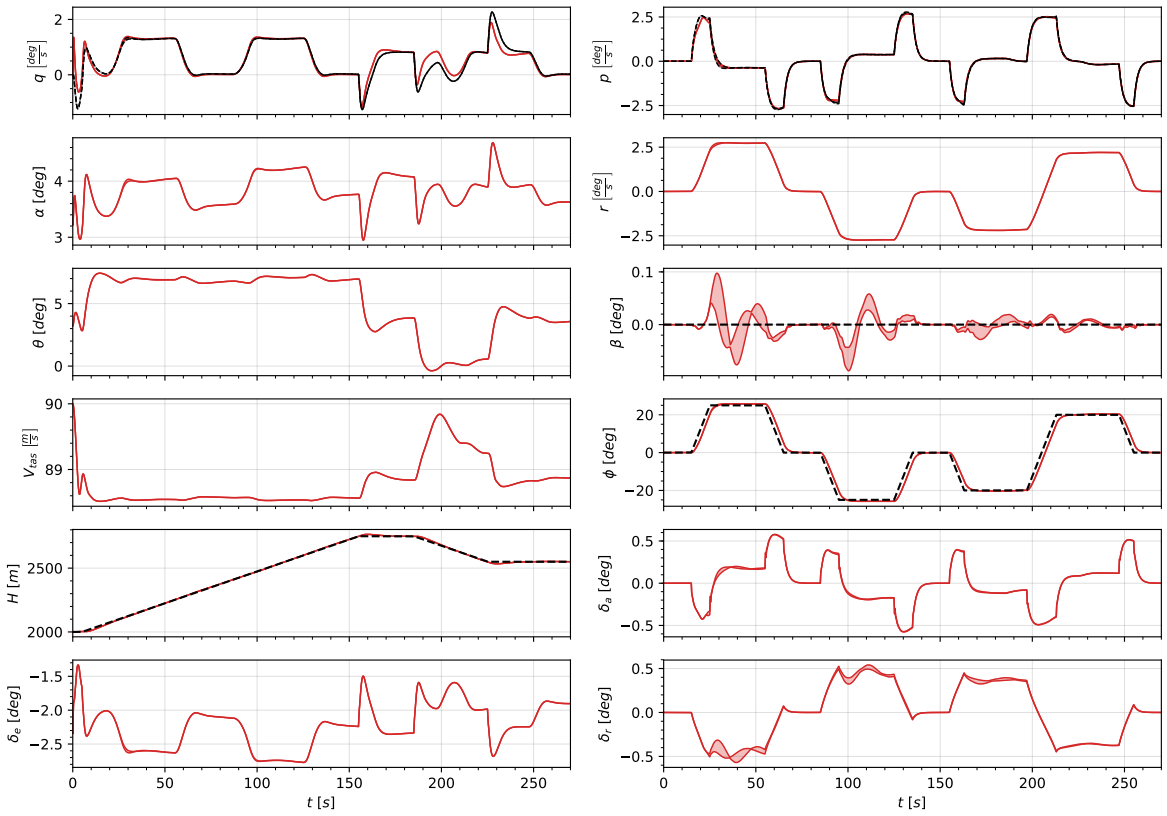

Figure 4.1: Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC0 $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines.

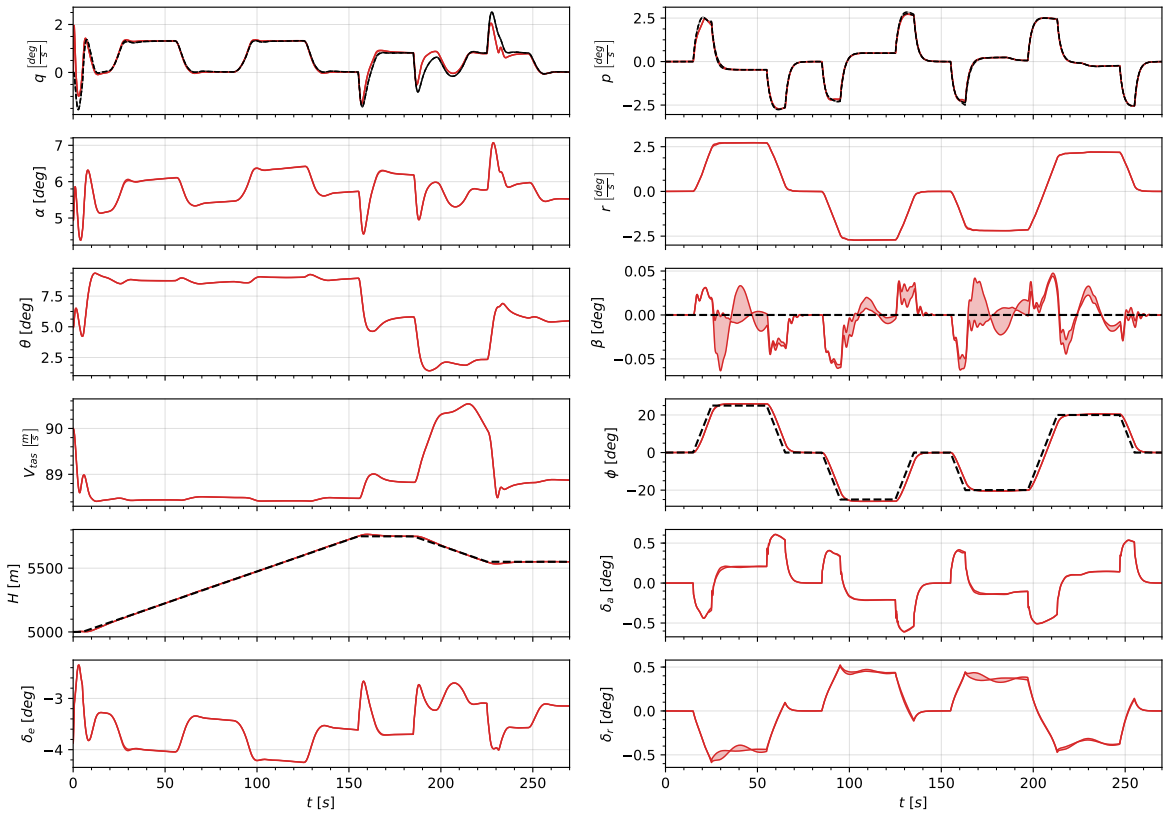

Figure 4.2: Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC2 $(V_{tas}, H) = (90\ \frac{m}{s}, 5\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines.

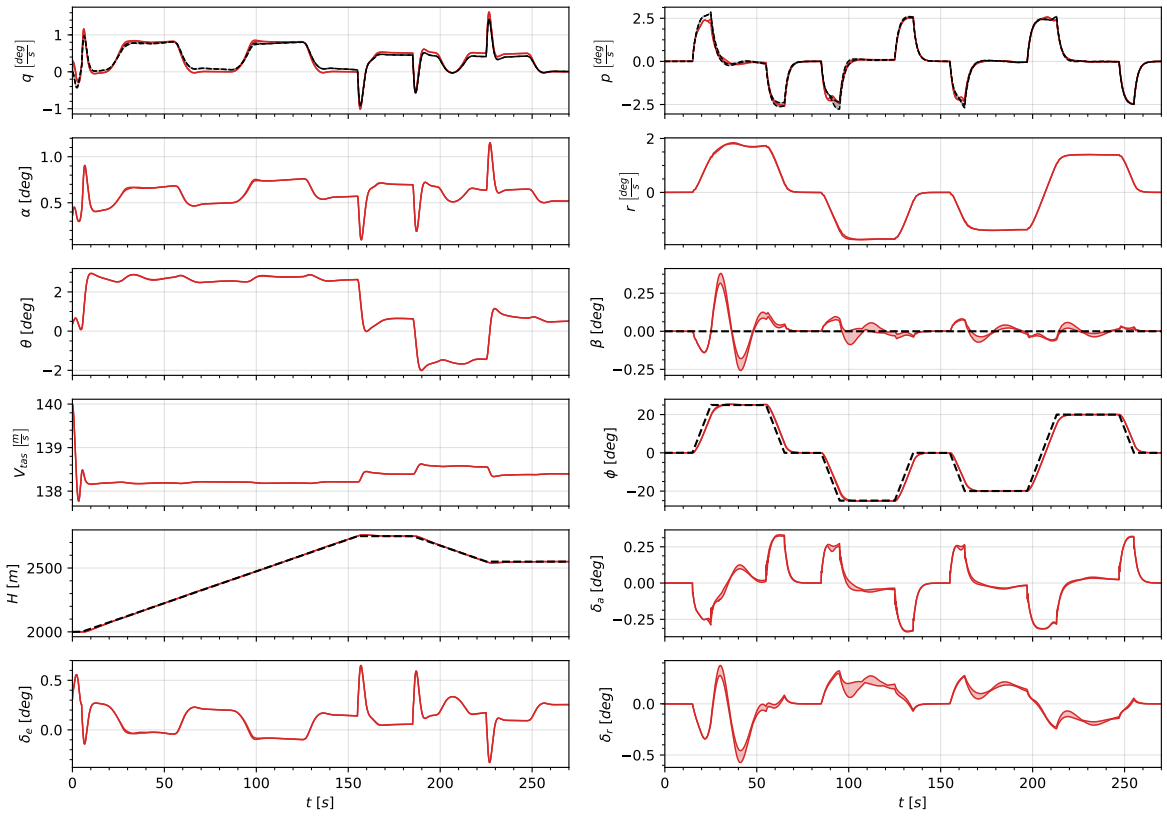

Figure 4.3: Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC1 $(V_{tas}, H) = (140\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines.

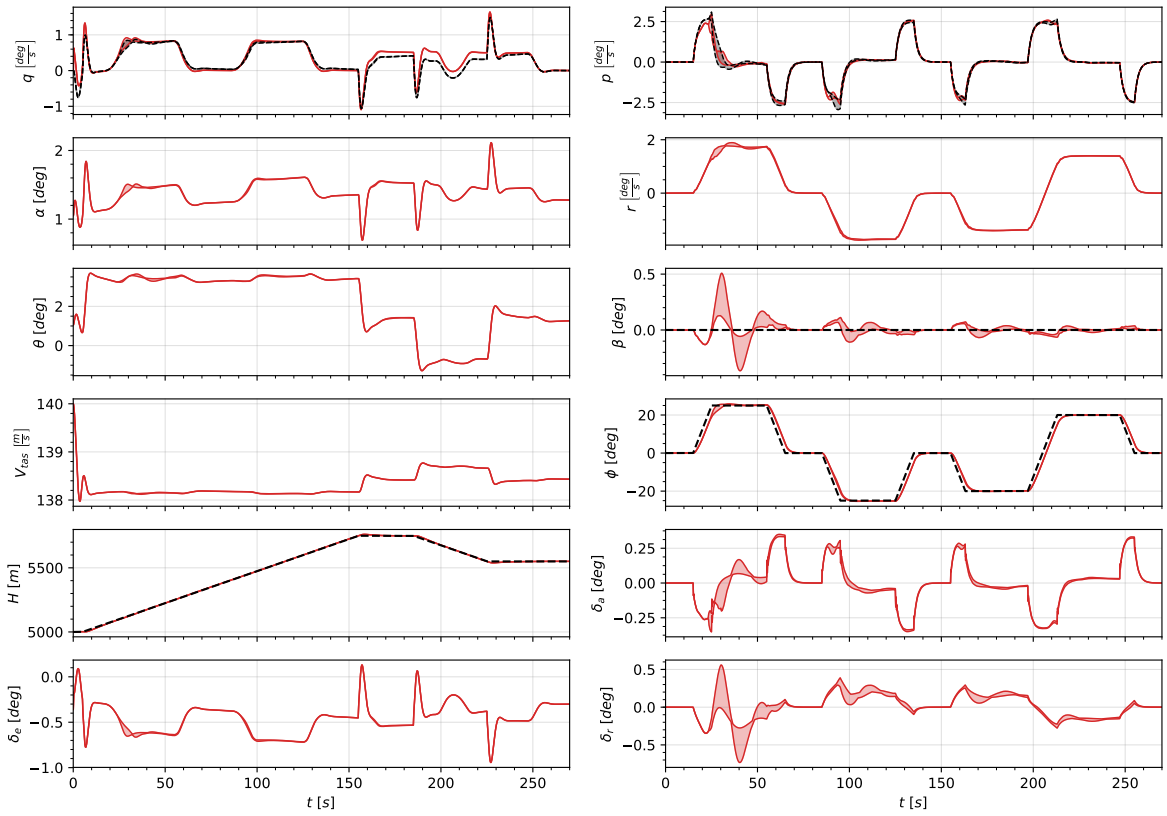

Figure 4.4: Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition FC3 $(V_{tas}, H) = (140\ \frac{m}{s}, 5\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines.

5

Simulation of Online Operation of a Large-Angle Maneuver

The maneuver flown in the previous chapter is representative for a normal operation of the PH-LAB aircraft. The maneuver consists of a prolonged climb and a subsequent descent (with $5\ \frac{m}{s}$), combined with left and right turns (of $25\ deg$ and $20\ deg$). Longitudinal-lateral-directional coupling effects are present during the majority of that maneuver, such as in the first 60 seconds, where the agent is required to perform a climbing steady turn, as depicted in Fig. 4.1. In this chapter, simulations are conducted for the same maneuver but with a larger magnitude for the climb/descent rate (with $10\ \frac{m}{s}$) and roll angle (of $50\ deg$ and $40\ deg$). These conditions are uncomfortable for passengers but could come about in an emergency situation. Although not a normal maneuver, these conditions are simulated to demonstrate the agent's performance in a scenario for which the nonlinear and coupling effects are significant. Furthermore, it becomes more difficult to coordinate the turn.

Figure 5.1 depicts the time histories of the minimum to maximum bound of 100 simulation runs. In all cases, it can be observed that the agent is able to fly the reference profile as commanded by the PID controllers. Hence, the agent does not fail in any of the 100 runs. Similarly to the results of the previous chapter, it can be observed that the agent keeps improving as it interacts with the environment. For example, the large sideslip angle as experienced during the first climbing right turn is reduced for the subsequent turns. Despite the more dynamic maneuver, with larger nonlinear and coupling effects, the agent with separate longitudinal and lateral controllers, is still able to control the aircraft.

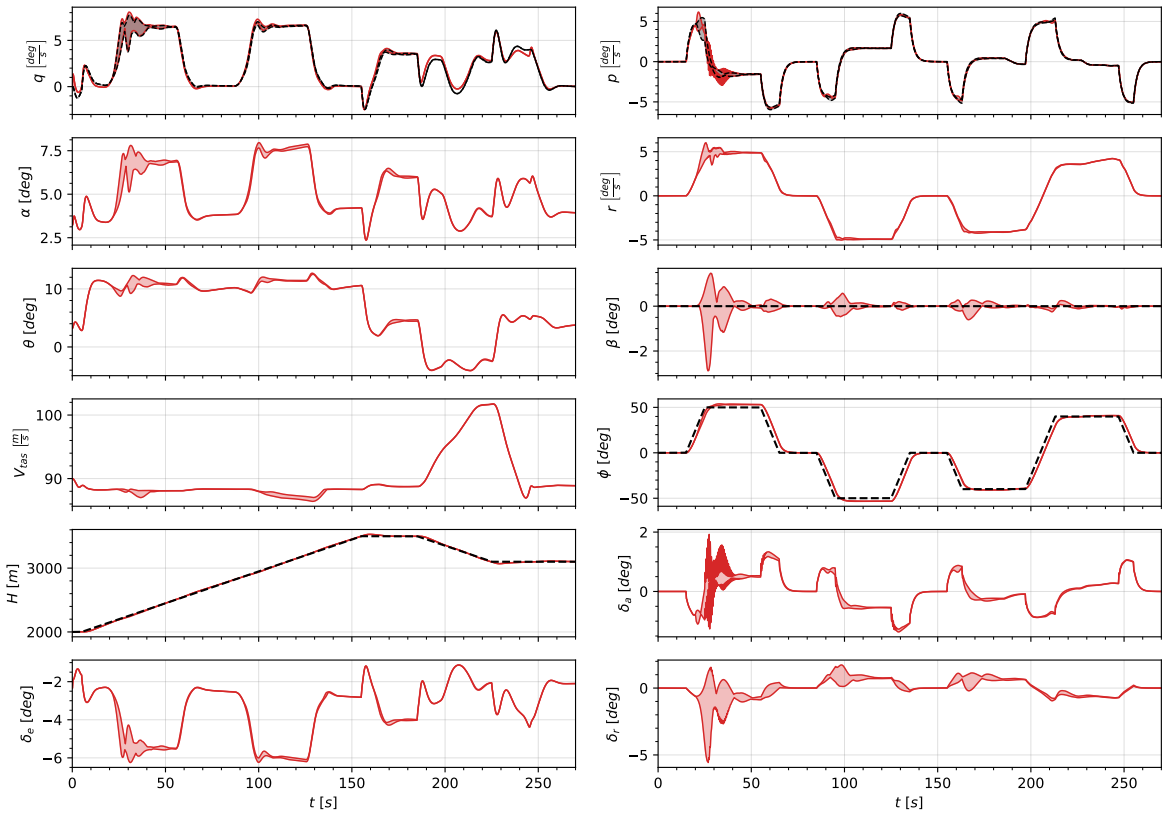

Figure 5.1: Min-max bounds over 100 runs of the online operation phase, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines.

6

Online System Identification Model Prediction Error

As presented in Part I, a Recursive Least Squares (RLS) estimator is utilized for online identification of the incremental model parameters, which play an important role in the update operations of both the actor and the critic. Most importantly, the model parameter estimates should have the correct sign and to some extent correct relative values. Reference values are derived from a plant linearization at the trim condition at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Figure 6.1 illustrates the online identified state and input matrices for both the longitudinal and lateral incremental models during the initial training phase, as discussed in Part I. In this chapter, the prediction error of the incremental model is presented and discussed for the online training phase. A comparison is made between the incremental model utilizing the parameters derived from the plant linearization and the online identified parameters.

Figure 6.2 illustrates the absolute state increment prediction error. The error for the incremental model with the parameters derived from the plant linearization is depicted in red. The incremental model with the online identified parameters in blue. For the sake of simplicity, the incremental model parameters acquired from the plant linearization are referred to as linear parameter from here on. From Fig. 6.2 it can be observed that with the exception of the sideslip angle increment, the model with the online identified parameters has a lower error than the model with linear parameters, after the initial 10 seconds. Furthermore, two trends can be observed. Whereas the error of the model with linear parameters increases over time, the error of the model with online identified parameters decreases with time. This is in line with the fact that as the aircraft departs from its initial trimmed state (at which the plant was linearized), the model with the linear parameters becomes less representative. Similarly, the model utilizing the online identified parameters improves with time as more data is acquired and converges to a final error value inherent to the representation capacity of the incremental model. The prediction error of the model with online identified states shows a quick decrease in the first 10 seconds. Only the prediction error of the sideslip angle is considerably slower. This behavior can be attributed to two things: (1) the initial incorrect identification of the $\frac{\partial p}{\partial \delta_r}$ term (2) decrease of the magnitude of the sideslip angles. The incorrect identification of the $\frac{\partial p}{\partial \delta_r}$ term, as illustrated in Fig, 6.1 originates from the indirect excitation of the rudder as elaborated in Part I. In addition as the lateral controller learns to minimize the sideslip angle during the training phase, less excitation of the sideslip angle is experienced by the agent, inhibiting the identification of that motion.

This is confirmed by the time history of sideslip angle prediction error in Fig. 6.3, where the error is computed as a percentage. The time history of the prediction error of the sideslip angle for both the model with linear and online identified parameters increases over time. This is a result of a decrease in the magnitude of the observed sideslip angle increment which leads to division by a small number. Similarly, the spikes in Fig. 6.3 happen at moments of local zero state increment. Last but not least, the online identified incremental model is a good representation of the plant as, with the exception of the sideslip angle, the prediction error is in general lower than 1%.

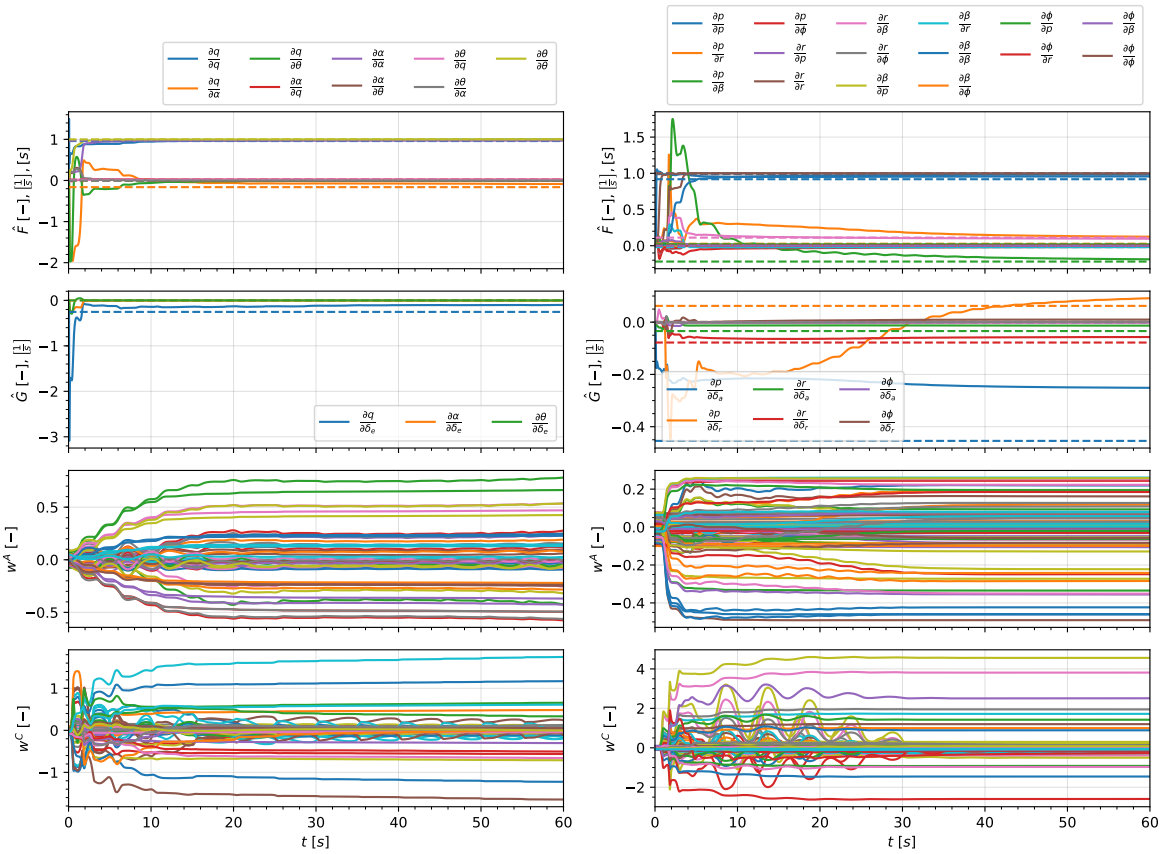

Figure 6.1: State and input matrix estimates, and actor and critic parameters during online training of longitudinal (left) and lateral (right) controller, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$ and $(\hat{\mathbf{F}}_0, \hat{\mathbf{G}}_0, \hat{\boldsymbol{\Theta}}_0) = (I, \mathbf{0}, I \cdot 10^8)$. Estimates derived from a plant linearization are represented by dotted lines.

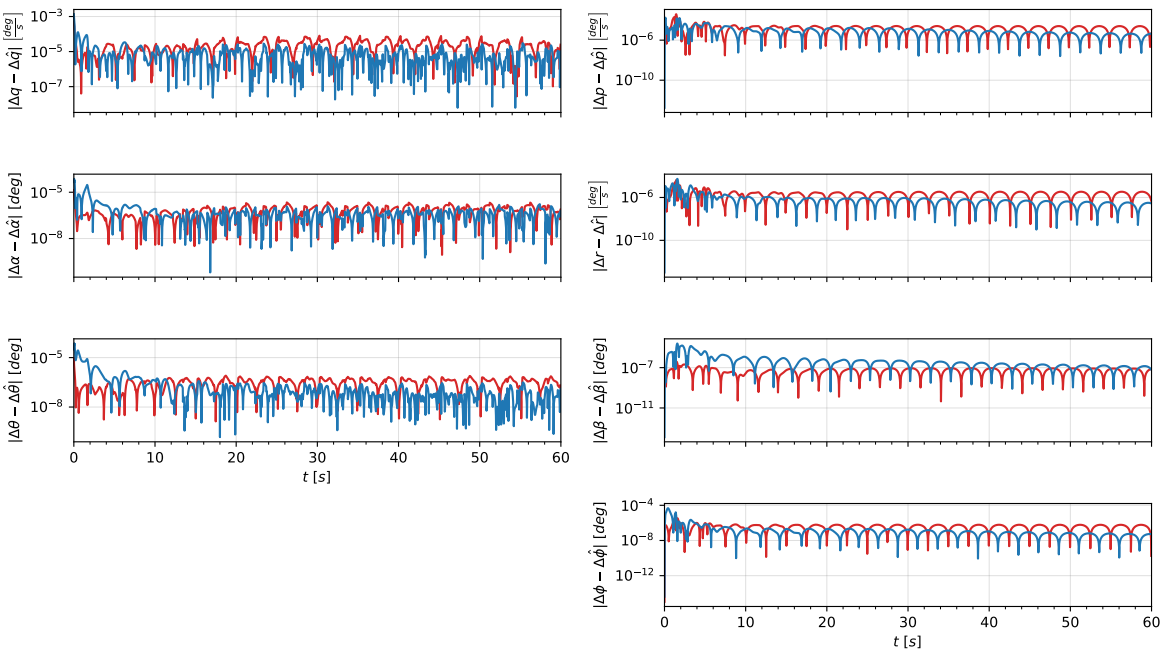

Figure 6.2: Absolute state increment prediction error of longitudinal (left) and lateral (right) controller and linear model, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$ and $(\hat{\mathbf{F}}_0, \hat{\mathbf{G}}_0, \hat{\mathbf{\Theta}}_0) = (I, \mathbf{0}, I \cdot 10^8)$. The prediction error of the incremental model using parameters from a plant linearization and from the online identification procedure are illustrated by red and blue line, respectively.

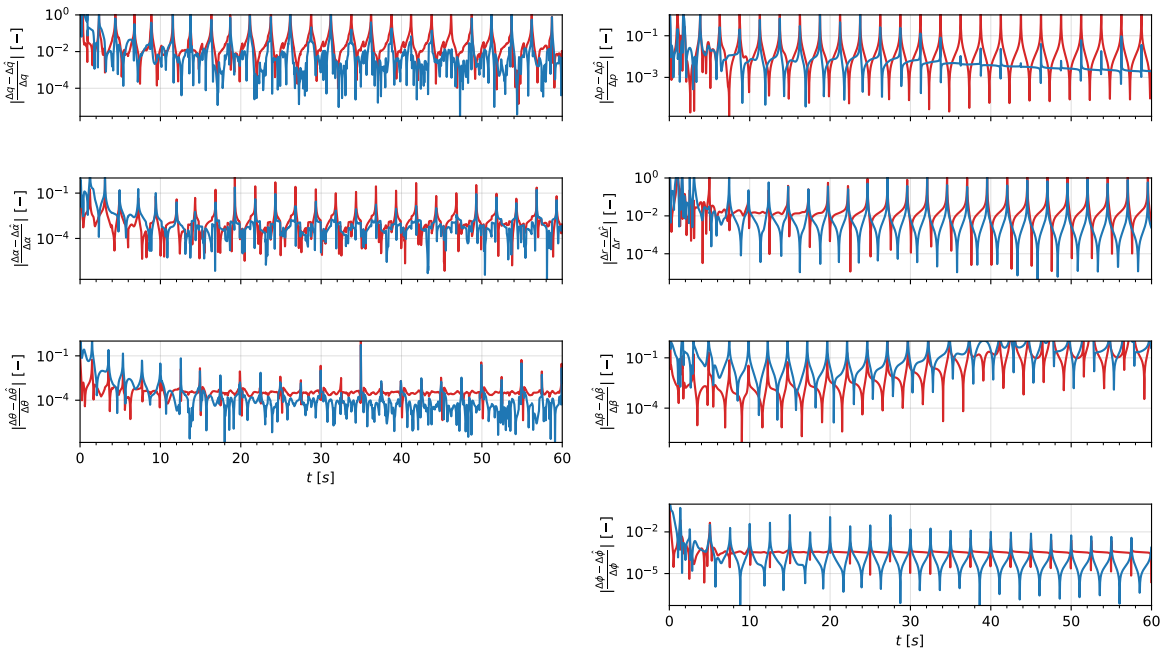

Figure 6.3: Absolute state increment prediction error percentage of longitudinal (left) and lateral (right) controller and linear model, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (1, 5, 10)$ and $(\hat{\mathbf{F}}_0, \hat{\mathbf{G}}_0, \hat{\mathbf{\Theta}}_0) = (I, \mathbf{0}, I \cdot 10^8)$. The prediction error of the incremental model using parameters from a plant linearization and from the online identification procedure are illustrated by red and blue line, respectively.

7

Estimator Covariance Windup

In the experiments conducted in the scientific paper, a forgetting factor of $\kappa = 1$ was utilized for the Recursive Least Squares (RLS) estimator to ensure its consistency. As elaborated in Part I, this is done to circumvent the effects of estimator windup during a period of poor or no excitation. In addition to poor excitation, non-uniformly distributed information over all parameters and time-scale separation in the variation of parameters also induce estimator windup. During estimator windup, the covariance matrix exponentially increases. Once, the system is excited again, abrupt changes in the estimated parameters occur despite no changes in the actual system. In this section, the goal is to demonstrate this effect by conducting an operation phase experiment, with a forgetting factor of $\kappa = 0.99$ for the online identification of both the longitudinal and lateral incremental models.

Figure 7.1 illustrates the performance of the agent during execution of the a flight profile initiated at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Figure 7.2 depicts the online identification process for the longitudinal and lateral incremental models during the flight. From Fig. 7.2 a saw-tooth pattern can be observed for the variance of the estimates. It shows that the variance exponentially (linearly in logarithmic scale) increases during the phases of the flight profile with no excitation. Once the system is excited again, aggressive changes in the estimates occur despite no changes in the actual system, as can be observed from the peaks in the state and input matrix estimates. As a consequence, the agent is provided with inconsistent plant information which on the long run can be detrimental to the controller's performance. From Fig. 7.1 it can be observed that the performance of the initially well-performing controller deteriorates, resulting in an increase of the sideslip angle during the execution of the maneuver.

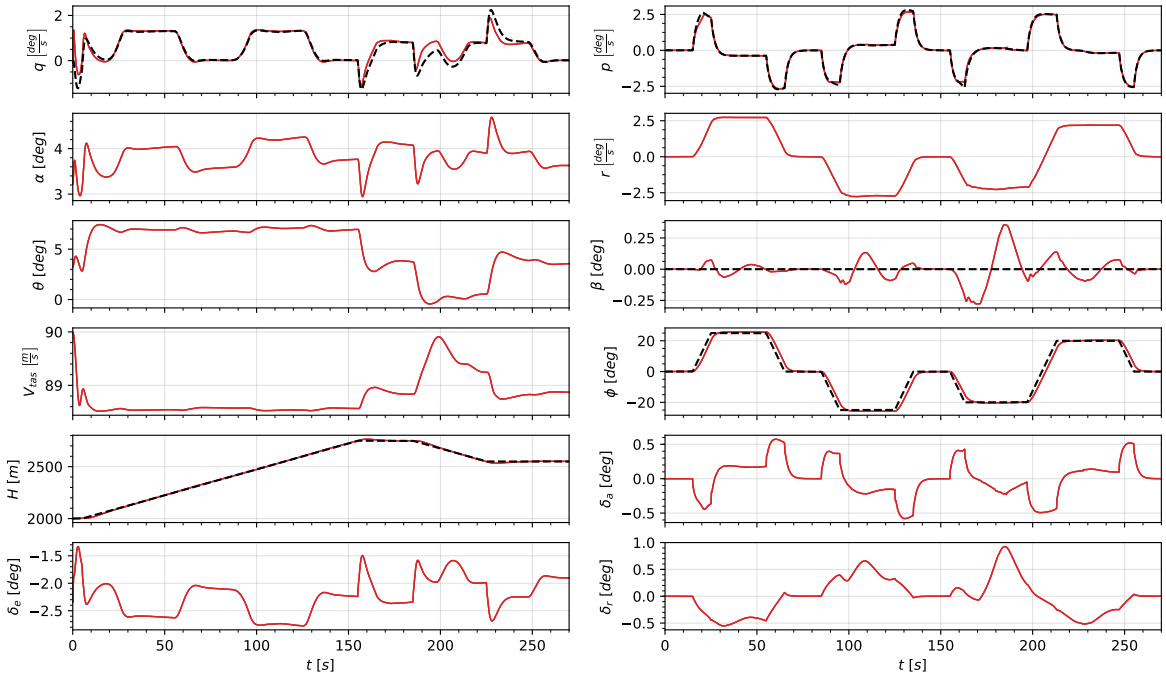

Figure 7.1: Online operation phase, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$ and pretrained agent, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. The agent was pretrained online at $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$. Reference signals are illustrated by black, dashed lines. A forgetting factor of $\kappa = 0.99$ is utilized for the online identification of both the longitudinal and lateral incremental models.

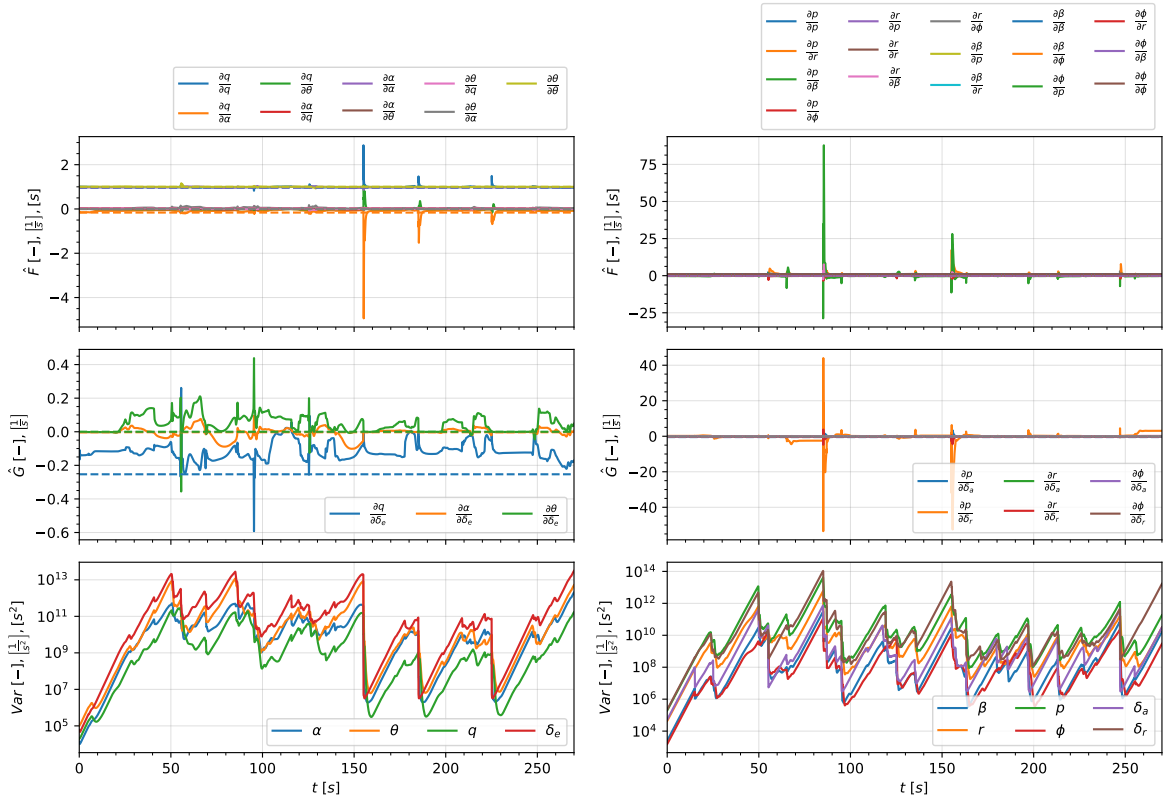

Figure 7.2: State and input matrix estimates, and variance during operation phase of longitudinal (left) and lateral (right) controller, starting at trimmed operation condition $(V_{tas}, H) = (90\ \frac{m}{s}, 2\ km)$, with $(\tau, \eta^A, \eta^C) = (0.01, 1, 2)$. Estimates derived from a plant linearization are represented by dotted lines. A forgetting factor of $\kappa = 0.99$ is utilized.

8

Verification and Validation

In this section, the verification and validation processes conducted throughout this thesis are reviewed and discussed. Verification is key to ensure the correct implementation of the derived Reinforcement Learning (RL) framework. Validation, on the other hand, allows for an evaluation of the representational power of the applied framework and models. Furthermore, the limitations posed by the assumptions made are identified.

As part of research goal (G2), as listed in Chapter 1, the reproducibility of the Incremental Dual Heuristic Programming (IDHP) framework is investigated in Chapter 3. The reproducibility and the correct implementation of the baseline framework are verified by applying the framework to a simplified, discrete-time, deterministic, Linear Time Invariant (LTI) short period model of the PH-LAB research aircraft as derived from [58]. This process is conducted in two stages. First, the agent, named Model Dependent Dual Heuristic Programming (MDDHP), is provided with the exact state and input matrices of the plant. This way, any possible issues introduced by an erroneous online system identification are eliminated, allowing for a verification of the actor's and critic's update rules. Subsequently, the MDDHP framework is augmented with the Recursive Least Squares (RLS) estimation of the incremental model, to allow for a verification of the complete IDHP framework. In both cases, the results from the tracking control experiments, as presented in Chapter 3, are consistent with the findings in [124].

As elaborated in Part I, a high-fidelity, nonlinear, six-degrees-of-freedom model of a Cessna 500 Citation I, developed with the Delft University Aircraft Simulation Model and Analysis Tool (DASMAT), is utilized for this research. Despite the differences of the aircraft in fuselage size, engine power, and wing size, the model is still representative of the PH-LAB research aircraft. The validity and representational power of the simulation model are studied in [104], where it is compared to flight data acquired with the PH-LAB research aircraft. It is shown that the relative root mean square error for the longitudinal force and moment coefficients equate to 9% and 13%, respectively. Similarly, the relative root mean squared error for the lateral force and moment coefficients is 7% and 9%, respectively. As a result, the simulation model is representative of the real PH-LAB research aircraft, throughout the normal, pre-stall flight envelope, in which the experiments in this thesis are conducted.

For assessing the validity of the simulation model it is important to emphasize the assumptions made in this research. In the experiments conducted in Part I, clean and synchronous measurements are assumed. Therewith measurement noise, delay, bias as well as quantization effects are neglected. These, however, can have an effect on both the learning process as well as the online system identification of the incremental model. Therefore an investigation of these effects is recommended in Part IV. Furthermore, in the design of the flight controller in Part I, the internal airspeed controller of the simulation model is utilized. In the Cessna 550 Citation II research aircraft however the airspeed is regulated by the pilot, as the aircraft does not have a dedicated auto-throttle function.

IV

Closure

9 Conclusion

The uncertainty and complexity of autonomous systems call for the development of advanced adaptive controllers. The proposed research aims to contribute to that goal, through the design and implementation of a Reinforcement Learning (RL) flight controller for the PH-LAB research aircraft, a Cessna 550 Citation II.

In Part II, the foundation of the thesis is established by reviewing published literature and conducting a preliminary analysis. Therefore, Part II focuses on answering the research question RQ1.

RQ1 Which RL algorithm is most applicable to the adaptive flight control of the PH-LAB research aircraft?

(RQ1.1) What are the current state-of-the-art RL algorithms for continuous control?

(RQ1.2) Is the proposed RL framework reproducible?

(RQ1.3) Does the proposed RL framework demonstrate satisfactory performance when applied to a simplified version of the PH-LAB research aircraft?

Following a short introduction to the fundamentals of RL, the literature survey focuses on approximate methods as these mitigate the curse of dimensionality for continuous systems such as the PH-LAB research aircraft. Methods that approximate the policy, inherently support both continuous state and action spaces and are therefore preferred. In combination with value approximation, the so-called actor-critic methods, have lower sample complexity and can be implemented online. Consequently, the literature scope is defined around actor-critic algorithms with nonlinear function approximation with Artificial Neural Networks (ANNs), which is well represented in the research fields of Approximate Dynamic Programming (ADP) and Deep Reinforcement Learning (DRL). From the subsequent review of the literature in both research fields it is concluded that, although ADP methods have a higher model-dependence than DRL, they are more suitable for an online learning application in the PH-LAB research aircraft due to there lower sample complexity. Nonetheless, common approaches for stabilization of the learning process, as introduced by DRL literature, should not be discarded as these can be applied to the learning process of ADP methods. The requirement of an initial offline training phase is the common limitation of ADP methods. Although it remains to be studied for complex, high-dimensional systems, an incremental approach could eliminate the need for an offline learning phase. Conclusively, the framework of Incremental Dual Heuristic Programming (IDHP) is proposed as a promising baseline RL framework for this research.

Subsequently, the proposed baseline framework of IDHP is reproduced and tested on a simple model of the PH-LAB research aircraft in the preliminary analysis in Chapter 3. After the derivation of the framework's update rules and training strategy, the preliminary experiments are conducted in two phases. In the fundamental testing phase, a Dual Heuristic Programming (DHP) agent is initially examined with exact model information and subsequently with an incremental model identified with as Recursive Least Squares (RLS). From the results, it is concluded that in both cases the agent is able to successfully learn a policy for the pitch rate tracking task. In the advanced testing phase, the

agent's performance is tested on different scenarios including, untrimmed initialization, measurement noise, and fault-tolerance. In all cases, the agent's performance remains satisfactory. Conclusively, the agent demonstrates good adaptability and robustness characteristics when applied to a simple model of the PH-LAB research aircraft. With the experiments conducted in the preliminary analysis, the characteristics of the IDHP framework are successfully reproduced and its applicability to the PH-LAB research aircraft confirmed.

In Part I of this thesis, the main body of the conducted research is presented through a scientific paper which answers the remaining research questions RQ2, RQ3, and RQ4.

RQ2 How can the proposed RL framework be incorporated into the flight controller design?

(RQ2.1) What are the dynamical system characteristics inherent to the PH-LAB research aircraft?

(RQ2.2) At which control level should the RL framework be embedded?

(RQ2.3) What are the implications of continual learning for the RL framework?

For this research, a high-fidelity, nonlinear, six-degrees-of-freedom model of a Cessna 500 Citation I, developed with the Delft University Aircraft Simulation Model and Analysis Tool (DASMAT) is utilized. Despite some differences to the PH-LAB research aircraft, the model is still representative [104]. The simulation model includes additional engine, actuator dynamics, and sensor models. The latter is not used as clean measurements are assumed. The actuator dynamics are modeled with a first order actuator model and control surface deflection saturation. Furthermore, the aircraft's yaw damper is disabled to provide the agent with full control authority over the aerodynamic control surfaces and the engine's thrust setting is controlled by an internal airspeed controller.

The adaptive learning framework is applied to angular rate control of the pitch and roll rate, augmented with an outer control loop. Rate control exhibits the lowest learning complexity due to the direct dynamic relation between the angular rates and control surfaces. The outer control loop consists of conventional PID controllers and provides a higher-level control interface to the flight profiles. Under the assumption of a symmetric aircraft and near symmetric flight conditions, a decoupled control design is employed utilizing separate longitudinal and lateral learning controllers. The airspeed is not provided to the longitudinal controller. The exclusion of the airspeed is motivated by the fact that the online identification of the airspeed related parameters of the incremental model is nontrivial due to time scale separation and relatively small local variations in airspeed. The latter is exacerbated by the internal airspeed controller. Additional to a reference roll rate, the lateral learning controller receives a zero degree reference for the sideslip angle, such that it can learn to minimize it, with the goal of flying coordinated turns.

As the learning framework continually learns during its operation it also experiences long periods with no excitation of the system. During periods of poor or no excitation, the covariance matrix of the RLS estimator of the incremental model increases exponentially. Once the system is excited again, abrupt changes in the estimate occur despite no changes in the actual system. As a consequence, the agent is provided with inconsistent plant information which on the long run can be detrimental to the controller's performance. An example of such an event is discussed in Part III, as part of the additional results. Conclusively, to ensure consistency of the estimator the forgetting factor of the RLS is set to one. Setting the forgetting factor to one has the disadvantage that the estimator does not forget older data. Consequently, the estimator becomes less adaptive over time. In this research, this problem is mitigated by resetting the covariance matrix to its initial value when a large change in the system is detected through the estimator's innovation term.

RQ3 How can the proposed RL framework be modified to improve controller adaptability and learning stability?

Unstable or diverging learning behavior leads to fast growth of the critic's parameters. As a consequence, the actor's loss gradients become very large leading to aggressive and large updates from which the agent cannot recover and fails. Consequently, an additional network structure, named target critic is introduced. The results presented in Part I are consistent with findings in [47], demonstrating that the target critic slows down and stabilizes the learning process. Especially in an application in

the PH-LAB aircraft, any failure is unacceptable. Conclusively, the augmentation of the current IDHP framework with the proposed target critic is vital to a successful implementation of the RL controller to the PH-LAB.

RQ4 What is the overall performance of the RL flight controller design?

(RQ4.1) What is the performance of the learning process measured in stability and speed?

(RQ4.2) How does it generalize to different flight regimes?

(RQ4.3) How does it perform during unexpected changes to the system or framework?

In Part I an online training procedure is presented. The proposed setup allows for a short training phase (under 60 seconds) with an acceptable load factor range. The results show that both the longitudinal and lateral controller is able to follow the reference signals after 30 seconds of training. In addition, training procedures are evaluated for untrimmed initial conditions. From the results, it is concluded that with the utilization of the proposed target critic, the failure rate during online training of the longitudinal controller decreases from 5.2% to 0%.

To demonstrate the agent's ability to operate the aircraft in different (not previously experienced) regimes, a maneuver is conducted for 4 different flight regimes. The representative maneuver consists of a prolonged climb and a subsequent descent (with $5\ \frac{m}{s}$), combined with left and right turns (of $25\ deg$ and $20\ deg$). The agent shows a failure rate of 0% in 100 runs simulated for each flight condition. In all cases, it can be observed that the agent is able to successfully fly the reference profile as commanded by the PID controllers. Furthermore, the conducted turns are well-coordinated as the sideslip is minimized, by the actuation of the rudder. In addition to the initial satisfactory control behavior, the agent keeps improving as it interacts with the environment.

The adaptability of the proposed framework is validated in two experiments. First, a failure is introduced to the aileron to demonstrate the framework's capability to quickly identify changes in the plant. Subsequently, another experiment is conducted where a disturbance is introduced directly in the actor's parameters. In the first experiment, the controller is able to detect the change in the plant and complete the turn, despite the failure in the ailerons. From the estimates of the input matrix of the lateral incremental model, it can be observed that the online identification perceives the change in the plant's dynamics. In the second experiment, the actor's parameters quickly adapt towards a new near-optimal policy within 10 seconds, after the injection of the disturbance.

Ultimately, the main goal MG of this research is reviewed.

MG Contribute to the development of *online*, *model-independent*, *adaptive* flight control for fixed wing aircraft with the purpose of dealing with *complex*, *continuous* dynamical systems, **by** *investigating* the applicability of novel Reinforcement Learning frameworks, through the *design* of a flight controller for the PH-LAB research aircraft.

In this thesis, the derivation, design, and analysis of a RL adaptive flight controller is presented and successfully implemented for a full-scale, high-fidelity aircraft simulation of the PH-LAB research aircraft. It is demonstrated that the proposed framework is able to learn a near-optimal control policy online without a priori knowledge of the system dynamics nor an offline training phase. Through the simulation of representative flight profiles, the results indicate that the learning controller is able to generalize and operate the aircraft in not previously encountered flight regimes as well as identify and adapt to unforeseen changes to the aircraft's dynamics. The framework proposed in this research mitigates the current limitation of Adaptive Critic Designs (ACDs) that require an offline training phase expanding the application of such algorithms to applications for which no accurate system model is available, nor readily identifiable.

Ultimately, the results of this research can accelerate the development of future aerial mobility concepts. The online and adaptive nature of the learning controller could enable fully autonomous operation in complex and dense urban environments, which is key to the economic feasibility of such concepts. The model independence of the framework is an added benefit, as representative models of novel Vertical Take-Off and Landing (VTOL) designs, proposed for these concepts, are not available. Last but not least, this thesis augments the current spectrum of adaptive flight control for passenger aircraft with a learning approach.

10

Recommendations

The proposed methods and results in this thesis provide the following insights for further research:

- One of the main characteristics inherent to an application to the PH-LAB research aircraft is continual learning. The learning process is not episodic but continues indefinitely. One of the many consequences of continual learning is the possibility of a monotonic increase in the magnitude of the parameters of the Artificial Neural Networks (ANNs). The increase in parameter magnitude could lead to the controller becoming more aggressive over time. In [125] the parameters are clipped after a predefined limit. This approach, however, is likely to freeze the agent, stopping the learning process. Therefore, the addition of parameter regularization, such as in [63], or the addition of an action penalty term to the reward function, such as in [27], should be investigated.

- In this research, the inputs of the actor and critic consist of the state vector and the reference tracking error vector. In contrary, in [23] the actor only has the reference tracking error vector as input, as an additional offline-trained trim network is utilized. It should be investigated if such a dedicated representation of the aircraft's trim map can be learned or identified online.

- For the online identification of the incremental model a Recursive Least Squares (RLS) estimator is utilized, with the assumption of clean measurements. The performance of the proposed framework should be investigated for noisy, delayed, asynchronous measurements. For that, the sensor model available in the high-fidelity simulation model of the PH-LAB [33] can be utilized.

- In this research, the learning framework is applied to angular rate control of the pitch and roll rate, as rate control exhibits the lowest learning complexity due to the direct dynamic relation between the angular rates and control surfaces. The extension of the learning framework to higher control levels and its effect on the learning process should be investigated. In addition, the application of cascaded network structures as proposed by [23, 125] could be beneficial.

- Under the assumption of a symmetric aircraft and near symmetric flight conditions this thesis employs a decoupled controller design with separate longitudinal and lateral learning controllers. A comparison should be conducted with a controller design where the longitudinal and lateral motion is controlled by a single controller, utilizing the same learning framework.

- For this research, the internal airspeed controller included in the simulation model of the PH-LAB research aircraft is utilized. An investigation of the learning framework should be conducted for the case where the controller has authority over the aircraft's thrust settings.

- The proposed learning framework in this thesis utilizes constant learning rates and discount factors. Scheduling of these parameters could improve the learning performance. Whereas in general, learning rates should be decreased over time, the discount factor should be increased. The latter encourages an initial myopic behavior of the agent, which can improve the stability of the learning process in its initial phase [86].

- During the operation in the PH-LAB research aircraft the agent experiences long periods of poor or no excitation. Poor excitation is the main contributor to estimator windup, which has a detrimental effect on the learning process. In this research, estimator consistency is ensured by setting the forgetting factor to one, which in turn decreases the agent's adaptability over time. Methods proposed in [15, 35, 41] as well as [66, 77] propose vector-type or selective forgetting, were individual parameter dependent forgetting factors are used. The application of such methods to the online system identification process of the incremental model should be investigated.

V

Appendices

A

Symmetric Equations of Motion

In Chapter 3 a simplified, discrete, longitudinal LTI model of the Cessna 550 Citation II is derived. The simplified model is derived from a more complex symmetric equations of motion [58], as defined by Equation (A.1). The definition of the symbols introduced in Equation (A.1) are presented in Table A.1. The numerical values for the coefficients of Table A.1 are presented in Table A.2.

$$\begin{bmatrix} \dot{\hat{u}} \\ \dot{\alpha} \\ \dot{\theta} \\ \frac{\dot{q}\bar{c}}{V} \end{bmatrix} = \begin{bmatrix} x_u & x_\alpha & x_\theta & 0 \\ z_u & z_\alpha & z_\theta & z_q \\ 0 & 0 & 0 & \frac{V}{\bar{c}} \\ m_u & m_\alpha & m_\theta & m_q \end{bmatrix} \begin{bmatrix} \hat{u} \\ \alpha \\ \theta \\ \frac{q\bar{c}}{V} \end{bmatrix} + \begin{bmatrix} x_{\delta_e} & x_{\delta_t} \\ z_{\delta_e} & z_{\delta_t} \\ 0 & 0 \\ m_{\delta_e} & m_{\delta_t} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \tag{A.1}$$

Table A.1: Symbols as utilized in Equation (A.1) [58].

	$x_{\dots}$	$z_{\dots}$	$m_{\dots}$
u	$\frac{V}{\bar{c}}\frac{C_{X_u}}{2\mu_c}$	$\frac{V}{\bar{c}}\frac{C_{Z_u}}{2\mu_c - C_{Z_{\dot{\alpha}}}}$	$\frac{V}{\bar{c}}\frac{C_{m_u} + C_{Z_u}\frac{C_{m_{\dot{\alpha}}}}{2\mu_c - C_{Z_{\dot{\alpha}}}}}{2\mu_c K_Y^2}$
α	$\frac{V}{\bar{c}}\frac{C_{X_\alpha}}{2\mu_c}$	$\frac{V}{\bar{c}}\frac{C_{Z_\alpha}}{2\mu_c - C_{Z_{\dot{\alpha}}}}$	$\frac{V}{\bar{c}}\frac{C_{m_\alpha} + C_{Z_\alpha}\frac{C_{m_{\dot{\alpha}}}}{2\mu_c - C_{Z_{\dot{\alpha}}}}}{2\mu_c K_Y^2}$
θ	$\frac{V}{\bar{c}}\frac{C_{Z_0}}{2\mu_c}$	$-\frac{V}{\bar{c}}\frac{C_{X_0}}{2\mu_c - C_{Z_{\dot{\alpha}}}}$	$-\frac{V}{\bar{c}}\frac{C_{X_0}\frac{C_{m_{\dot{\alpha}}}}{2\mu_c - C_{Z_{\dot{\alpha}}}}}{2\mu_c K_Y^2}$
q	$\frac{V}{\bar{c}}\frac{C_{X_q}}{2\mu_c}$	$\frac{V}{\bar{c}}\frac{2\mu_c + C_{Z_q}}{2\mu_c - C_{Z_{\dot{\alpha}}}}$	$\frac{V}{\bar{c}}\frac{C_{m_q} + C_{m_{\dot{\alpha}}}\frac{2\mu_c + C_{Z_q}}{2\mu_c - C_{Z_{\dot{\alpha}}}}}{2\mu_c K_Y^2}$
δ_e	$\frac{V}{\bar{c}}\frac{C_{X_{\delta_e}}}{2\mu_c}$	$\frac{V}{\bar{c}}\frac{C_{Z_{\delta_e}}}{2\mu_c - C_{Z_{\dot{\alpha}}}}$	$\frac{V}{\bar{c}}\frac{C_{m_{\delta_e}} + C_{Z_{\delta_e}}\frac{C_{m_{\dot{\alpha}}}}{2\mu_c - C_{Z_{\dot{\alpha}}}}}{2\mu_c K_Y^2}$
δ_t	$\frac{V}{\bar{c}}\frac{C_{X_{\delta_t}}}{2\mu_c}$	$\frac{V}{\bar{c}}\frac{C_{Z_{\delta_t}}}{2\mu_c - C_{Z_{\dot{\alpha}}}}$	$\frac{V}{\bar{c}}\frac{C_{m_{\delta_t}} + C_{Z_{\delta_t}}\frac{C_{m_{\dot{\alpha}}}}{2\mu_c - C_{Z_{\dot{\alpha}}}}}{2\mu_c K_Y^2}$

Table A.2: Symmetric stability and control derivatives [58].

V	=	59.9 m/sec	m	=	4547.8 kg	$\bar{c}$	= 2.022 m
S	=	24.2 m^2	l_h	=	5.5 m	μ_c	= 102.7
K_Y^2	=	0.980	x_{cg}	=	0.30 $\bar{c}$		
C_{X_0}	=	0	C_{Z_0}	=	−1.1360		
C_{X_u}	=	−0.2199	C_{Z_u}	=	−2.2720	C_{m_u}	= 0
C_{X_α}	=	0.4653	C_{Z_α}	=	−5.1600	C_{m_α}	= −0.4300
$C_{X_{\dot{\alpha}}}$	=	0	$C_{Z_{\dot{\alpha}}}$	=	−1.4300	$C_{m_{\dot{\alpha}}}$	= −3.7000
C_{X_q}	=	0	C_{Z_q}	=	−3.8600	C_{m_q}	= −7.0400
C_{X_δ}	=	0	C_{Z_δ}	=	−0.6238	C_{m_δ}	= −1.5530

Bibliography

[1] P. Acquatella, W. Falkena, E. Van Kampen, and Q. P. Chu. Robust Nonlinear Spacecraft Attitude Control using Incremental Nonlinear Dynamic Inversion. In *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, Minnesota, 8 2012.

[2] P. Acquatella, E. van Kampen, and Q. P. Chu. Incremental Backstepping for Robust Nonlinear Flight Control. In *Proceedings of the EuroGNC*, Delft, The Netherlands, 2013.

[3] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight Experience Replay. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5048–5058. Curran Associates, Inc., 2017.

[4] A. Antos, C. Szepesvári, and R. Munos. Fitted Q-iteration in continuous action-space MDPs. In J C Platt, D Koller, Y Singer, and S T Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 9–16. Curran Associates, Inc., 2008.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 11 2017.

[6] S. N. Balakrishnan and V. Biega. Adaptive-Critic-Based Neural Networks for Aircraft Optimal Control. *Journal of Guidance, Control, and Dynamics*, 19(4):893–898, 1996.

[7] G. J. Balas. Flight Control Law Design: An Industry Perspective. *European Journal of Control*, 9(2-3):207–226, 4 2003.

[8] N. E. Barabanov and D. V. Prokhorov. Stability Analysis of Discrete-Time Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 13(2):292–303, 2002.

[9] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, A. Muldal, N. Heess, and T. Lillicrap. Distributed Distributional Deterministic Policy Gradients. In *International Conference on Learning Representations (ICLR)*, 2018.

[10] 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*, 13(5): 834–846, 1983.

[11] J. A. J. Berni, P. J. Zarco-Tejada, L. Suárez, and E. Fereres. Thermal and Narrowband Multispectral Remote Sensing for Vegetation Monitoring From an Unmanned Aerial Vehicle. *IEEE Transactions on Geoscience and Remote Sensing*, 47(2):722–738, 3 2009.

[12] D. P. Bertsekas, M. L. Homer, D. A. Logan, S. D. Patek, and N. R. Sandell. Missile defense and interceptor allocation by neuro-dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(1):42–51, 1 2000.

[13] J. A. Boyan. Technical Update: Least-Squares Temporal Difference Learning. *Machine Learning*, 49(2):233–246, 11 2002.

[14] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC press, 2010.

[15] L. Cao and H. M. Schwartz. A novel recursive algorithm for directional forgetting. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 2, pages 1334–1338. IEEE, 1999.

[16] R. R. da Costa, Q. P. Chu, and J. A. Mulder. Reentry Flight Controller Design Using Nonlinear Dynamic Inversion. *Journal of Spacecraft and Rockets*, 40(1):64–71, 1 2003.

[17] C. Darken, J. Chang, and J. Moody. Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, pages 3–12, Helsingoer, 1992. IEEE.

[18] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 27*, pages 2933–2941, 2014.

[19] P. Dayan. The convergence of TD(λ) for general λ. *Machine learning*, 8(3-4):341–362, 1992.

[20] P. Dayan and T. J. Sejnowski. TD(λ) converges with probability 1. *Machine Learning*, 14(3): 295–301, 1994.

[21] R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence Innovative Applications of Artificial Intelligence*, pages 761–768. American Association for Artificial Intelligence, 1998.

[22] J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[23] R. Enns and J. Si. Helicopter trimming and tracking control using direct neural dynamic programming. *IEEE Transactions on Neural Networks*, 14(4):929–939, 7 2003.

[24] J. Farrell, M. Sharma, and M. Polycarpou. Backstepping-Based Flight Control with Adaptive Function Approximation. *Journal of Guidance, Control, and Dynamics*, 28(6):1089–1102, 11 2005.

[25] S Ferrari and R F Stengel. Algebraic training of a neural network. In *Proceedings of the American Control Conference*, volume 2, pages 1605–1610, 2001.

[26] S. Ferrari and R. F. Stengel. Classical/Neural Synthesis of Nonlinear Control Systems. *Journal of Guidance, Control, and Dynamics*, 25(3):442–448, 2002.

[27] Silvia Ferrari and Robert F. Stengel. Online Adaptive Critic Flight Control. *Journal of Guidance, Control, and Dynamics*, 27(5):777–786, 9 2004.

[28] H. Freimuth, J. Müller, and M. König. Simulating and executing UAV-assisted inspections on construction sites. In *Proceedings of the 34th International Symposium on Automation and Robotics in Construction (ISARC)*, pages 647–654, 2017.

[29] S. Fujimoto, H. van Hoof, and D. Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *35th International Conference on Machine Learning (ICML)*, 2018.

[30] A. Geramifard, M. Bowling, and R. S. Sutton. Incremental least-squares temporal difference learning. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 1, pages 356–361. AAAI Press, 2006.

[31] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, pages 850–855, Edinburgh, 1999. IET.

[32] E. Greensmith, P. L. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.

[33] F. Grondman, G. Looye, R. O. Kuchar, Q. P. Chu, and E. van Kampen. Design and Flight Testing of Incremental Nonlinear Dynamic Inversion-based Control Laws for a Passenger Aircraft. In *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida, 1 2018.

[34] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[35] T. Hägglund. Recursive Estimation of Slowly Time-Varying Parameters. *IFAC Proceedings Volumes*, 18(5):1137–1142, 7 1985.

[36] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia, 1995.

[37] S. Kim, Y. Kim, and C Song. A robust adaptive nonlinear control approach to missile autopilot design. *Control Engineering Practice*, 12(2):149–154, 2004.

[38] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 12 2014.

[39] V. Klein and E. A. Morelli. *Aircraft System Identification: Theory and Practice*. American Institute of Aeronautics and Astronautics, 2006.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2013.

[41] R. Kulhavý. Restricted exponential forgetting in real-time identification. *Automatica*, 23(5):589–600, 9 1987.

[42] S. Kumar, J. Harding, and S. Bass. AH-64 Apache Engineering Simulation Non-Real Time Validation Manual. Technical report, USAAVSCOM TR 90-A-010, 1990.

[43] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[44] S. H. Lane and R. F. Stengel. Flight control design using non-linear inverse dynamics. *Automatica*, 24(4):471–483, 7 1988.

[45] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[46] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2017.

[47] 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. In *International Conference on Learning Representations (ICLR)*, 2016.

[48] L. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[49] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li. *Adaptive dynamic programming with applications in optimal control*. Springer, 2017.

[50] Y. Liu, M. Kreimeier, E. Stumpf, Y. Zhou, and H. Liu. Overview of recent endeavors on personal aerial vehicles: A focus on the US and Europe led research activities. *Progress in Aerospace Sciences*, 91:53–66, 5 2017.

[51] T. Lombaerts, E. V. Oort, Q. P. Chu, J. A. Mulder, and D. Joosten. Online Aerodynamic Model Structure Selection and Parameter Estimation for Fault Tolerant Control. *Journal of Guidance, Control, and Dynamics*, 33(3):707–723, 5 2010.

[52] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pages 6379–6390, Long Beach, CA, USA, 2017. Curran Associates, Inc.

[53] P. Lu, E. van Kampen, C. de Visser, and Q. P. Chu. Aircraft fault-tolerant trajectory control using Incremental Nonlinear Dynamic Inversion. *Control Engineering Practice*, 57:126–141, 12 2016.

[54] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control*, 46:191–209, 2001.

[55] W. T. Miller, P. J. Werbos, and R. S. Sutton. *Neural networks for control*. MIT press, 1995.

[56] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2 2015.

[57] 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 *33rd International Conference on Machine Learning (ICML)*, volume 48, pages 1928–1937, New York, NY, USA, 2016.

[58] J. A. Mulder, W. H. J. J. van Staveren, J. C. van der Vaart, E. de Weerdt, C. C. de Visser, A. C. in 't Veld, and E. Mooij. *Flight Dynamics: Lecture Notes AE3202*. TU Delft, Aerospace Engineering Faculty, Delft, 2013.

[59] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2016.

[60] A. Nedić and D. P. Bertsekas. Least Squares Policy Evaluation Algorithms with Linear Function Approximation. *Discrete Event Dynamic Systems*, 13(1-2):79–110, 1 2003.

[61] Y. Nesterov. A Method for Solving a Convex Programming Problem with Convergence Rate O(1/k^2). *Soviet Mathematics Doklady*, 27:372–367, 1983.

[62] F. Nex and F. Remondino. UAV for 3D mapping applications: a review. *Applied Geomatics*, 6(1): 1–15, 3 2014.

[63] A. Y. Ng. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA, 2004. ACM.

[64] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[65] Z. Ni, H. He, X. Zhong, and D. V. Prokhorov. Model-Free Dual Heuristic Dynamic Programming. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8):1834–1839, 2015.

[66] J. E. Parkum, N. K. Poulsen, and J. Holst. Selective Forgetting in Adaptive Procedures. *IFAC Proceedings Volumes*, 23(8):137–142, 8 1990.

[67] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[68] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *arXiv preprint arXiv:1802.09464*, 2 2018.

[69] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 697–704. ACM, 2006.

[70] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2007.

[71] D. V. Prokhorov and D. C. Wunsch. Adaptive critic designs. *IEEE Transactions on Neural Networks*, 8(5):997–1007, 1997.

[72] I. E. Putro and F. Holzapfel. Robust flight control design using incremental adaptive sliding mode control. In *International Conference on Instrumentation, Control and Automation*, pages 114–119, 2016.

[73] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12 (1):145–151, 1 1999.

[74] M. Riedmiller. Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In João Gama, Rui Camacho, Pavel B Brazdil, Alípio Mário Jorge, and Luís Torgo, editors, *Machine Learning: ECML 2005*, pages 317–328, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[75] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 3rd edition, 2016.

[76] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[77] S. Saelid and B. Foss. Adaptive controllers with a vector variable forgetting factor. In *The 22nd IEEE Conference on Decision and Control*, pages 1488–1494. IEEE, 1983.

[78] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*, 2016.

[79] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

[80] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

[81] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[82] W. Schultz, P. Dayan, and P. R. Montague. A Neural Substrate of Prediction and Reward. *Science*, 275(5306):1593–1599, 3 1997.

[83] M. Sghairi, A. Bonneval, Y. Crouzet, J. J. Aubert, and P. Brot. Challenges in Building Fault-Tolerant Flight Control System for a Civil Aircraft. *IAENG International Journal of Computer Science*, 35(4), 1 2008.

[84] J. S. Shamma and M. Athans. Guaranteed Properties of Gain Scheduled Control for Linear Parameter-Varying Plants. *Automatica*, 27(3):559–564, 1991.

[85] J. Si and Y. Wang. Online learning control by association and reinforcement. *IEEE Transactions on Neural Networks*, 12(2):264–276, 2001.

[86] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch. *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press, 2004.

[87] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. *Journal of Guidance, Control, and Dynamics*, 33(6):1732–1742, 11 2010.

[88] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

[89] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 1 2016.

[90] P Simplício, M. D. Pavel, E. van Kampen, and Q. P. Chu. An acceleration measurements-based approach for helicopter nonlinear flight control using incremental Nonlinear Dynamic Inversion. *Control Engineering Practice*, 21(8):1065–1077, 8 2013.

[91] E. J. J. Smeur, Q. P. Chu, and G. C. H. E. de Croon. Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 3 2016.

[92] L. Sonneveldt, Q. P. Chu, and J. A. Mulder. Nonlinear Flight Control Design Using Constrained Adaptive Backstepping. *Journal of Guidance, Control, and Dynamics*, 30(2):322–336, 3 2007.

[93] L. Sonneveldt, E. R. van Oort, Q. P. Chu, and J. A. Mulder. Comparison of Inverse Optimal and Tuning Functions Designs for Adaptive Missile Control. *Journal of Guidance, Control, and Dynamics*, 31(4):1176–1182, 7 2008.

[94] L. Sonneveldt, E. R. van Oort, Q. P. Chu, C. C. De Visser, J. A. Mulder, and J. H. Breeman. Lyapunov-based Fault Tolerant Flight Control Designs for a Modern Fighter Aircraft Model. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Chicago, Illinois, 8 2009.

[95] L. Sonneveldt, E. R. van Oort, Q. P. Chu, and J. A. Mulder. Nonlinear Adaptive Trajectory Control Applied to an F-16 Model. *Journal of Guidance, Control, and Dynamics*, 32(1):25–39, 1 2009.

[96] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[97] M. Srouji, J. Zhang, and R. Salakhutdinov. Structured Control Nets for Deep Reinforcement Learning. *arXiv preprint arXiv:1802.08311*, 2 2018.

[98] 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*, pages 881–888. ACM, 2006.

[99] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient natural evolution strategies. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 539–546, Montreal, Québec, Canada, 2009. ACM.

[100] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. A Bradford Book, 2nd edition, 2018.

[101] 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 (NIPS)*, volume 12, pages 1057–1063, 2000.

[102] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

[103] J. N. Tsitsiklis and B. Van Roy. An Analysis of Temporal-Difference Learning with Function Approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

[104] M. A. van den Hoek, C. C. de Visser, and D. M. Pool. Identification of a Cessna Citation II Model Based on Flight Test Data. In *Advances in Aerospace Guidance, Navigation and Control*, pages 259–277, Cham, 2018. Springer International Publishing.

[105] H. van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, volume 23, pages 2613–2621. The MIT Press, 2010.

[106] H. van Hasselt, A. Guez, and D. Silver. Deep Reinforcement Learning with Double Q-Learning. In *30th AAAI Conference on Artificial Intelligence*, pages 2094–2100, Phoenix, Arizona, US, 2016. AAAI Press.

[107] E. van Kampen, Q. P. Chu, and J. A. Mulder. Continuous Adaptive Critic Flight Control Aided with Approximated Plant Dynamics. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Reston, Virigina, 8 2006. American Institute of Aeronautics and Astronautics.

[108] E. R. van Oort, L. Sonneveldt, Q. P. Chu, and J. A. Mulder. Full-Envelope Modular Adaptive Control of a Fighter Aircraft Using Orthogonal Least Squares. *Journal of Guidance, Control, and Dynamics*, 33(5):1461–1472, 9 2010.

[109] H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering. A theoretical and empirical analysis of Expected Sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184. IEEE Press, 2009.

[110] 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(3):764–773, 2002.

[111] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin. Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica*, 48(8):1825–1832, 2012.

[112] X. Wang, E. van Kampen, Q. P. Chu, and P. Lu. Stability Analysis for Incremental Nonlinear Dynamic Inversion Control. In *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida, 1 2018.

[113] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *International Conference on Learning Representations (ICLR)*, 2017.

[114] C. J. C. H. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 8(3):279–292, 1992.

[115] M. Wiering and M. van Otterlo. *Reinforcement Learning: State-of-the-Art*. Springer-Verlag Berlin Heidelberg, 2012.

[116] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[117] I. H. Witten. An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, 34(4):286–295, 1977.

[118] D. D. Woods. The risks of autonomy: Doyle's catch. *Journal of Cognitive Engineering and Decision Making*, 10(2):131–133, 6 2016.

[119] Y. Wu, E. Mansimov, S. Liao, A. Radford, and J. Schulman. OpenAI Baselines: ACKTR & A2C, 2017.

[120] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 12 2012.

[121] Y. Zhou, E. van Kampen, and Q. P. Chu. Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control. In *International Micro Air Vehicle Conference and Competition 2016*, pages 173–180, Beijing, 2016.

[122] Y. Zhou, E. van Kampen, and Q. P. Chu. Nonlinear Adaptive Flight Control Using Incremental Approximate Dynamic Programming and Output Feedback. *Journal of Guidance, Control, and Dynamics*, 40(2):493–496, 2 2017.

[123] Y. Zhou, E. van Kampen, and Q. P. Chu. Launch Vehicle Adaptive Flight Control with Incremental Model Based Heuristic Dynamic Programming. In *68th International Astronautical Congress (IAC)*, Adelaide, Australia, 2017.

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

[125] Y. Zhou, E. van Kampen, and Q. P. Chu. Incremental Approximate Dynamic Programming for Nonlinear Adaptive Tracking Control with Partial Observability. *Journal of Guidance, Control, and Dynamics*, 41(12):2554–2567, 12 2018.