Safe Reinforcement Learning in Flight Control

Introduction to Safe Incremental Dual Heuristic Programming

R.R. Feith

Delft, University of Technology

TU Delft

Safe Reinforcement Learning in Flight Control

Introduction to Safe Incremental Dual Heuristic Programming

by

R.R. Feith

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday January 30, 2020 at 14:00.

Student number:	4218272	
Project duration:	January 2019 – January 2020	
Thesis committee:	Dr. P. Chu,	TU Delft, Chair
	Dr. ir. E. van Kampen,	TU Delft, Daily supervisor
	Dr. ir. D. Dirkx,	TU Delft, External examiner
	MSc. PhD-student B. Sun,	TU Delft

TUDelft

Preface

”Today’s action hero, his skills are through technology. He can fly...”
-Sylvester Stallone

The completion of this thesis also marks the completion of an era and the dawning of new adventures. Let this work be the cherry on the pie. I dedicate this work to my wife, Yareena, who has supported me from start to finish with kind words, everlasting faith and unending love. Furthermore I also want to thank my parents for encouraging me to build a future through education and supporting and loving me along the way.

I would also like to thank my daily supervisor, Erik-Jan, who has tutored me throughout this research. You taught me to to translate ideas into designs and designs into reality. Your encouragement always boosted my confidence.

Lastly I would like to thank some of my friends and fellow students who offered both friendship and feedback when required, thank you Bas and Leo.

R.R. Feith
Delft, January 2020

Contents

List of Figures

List of Tables

Nomenclature

s	=	state vector
a_t	=	action vector
a_p	=	proposed action vector
a_c	=	corrective action vector
a_s	=	safe action vector
L^a	=	actor loss function
L^C	=	critic loss function
A	=	amplitude
f	=	frequency
r	=	reward
t	=	current timestep
W^C	=	critic neural network weights
W^A	=	actor neural network weights
η^C	=	critic learning rate
η^A	=	actor learning rate
η^s	=	safe learning rate multiplier
$F, \hat{F}$	=	state matrix, estimated state matrix
$G, \hat{G}$	=	input matrix, estimated input matrix
γ	=	discount factor
e	=	error vector
$v(s)$	=	value function
$q_\pi(s, a), Q(s, a)$	=	action-value function
$A(s, a)$	=	advantage function
$J(\cdot)$	=	cost function
G	=	cumulative reward
C	=	constraint
$\lambda(s)$	=	partial derivative of the value function with respect to each state
$\pi(s)$	=	policy function
$\mathbb{E}[\cdot]$	=	expectation of a random variable
$p(x\|y)$	=	probability of x given y
δ	=	temporal-difference error
Θ	=	RLS weights
Λ	=	RLS covariance matrix
X	=	RLS measurement matrix
ϵ	=	RLS prediction error
k	=	RLS forgetting factor
n	=	load factor
q	=	pitch rate
α	=	angle of attack
θ	=	pitch angle
γ	=	flight path angle
V	=	Velocity
H	=	Height
δ_e	=	elevator deflection

Acronyms

ANN	=	Artifical Neural Network
DDPG	=	Deep Deterministic Policy Gradient
NAF	=	Normalized Advantage Function
PPO	=	Proximal Policy Optimization
DHP	=	Dual Heuristic Programming
IDHP	=	Incremental model based Dual Heuristic Programming
SIDHP	=	Safe Incremental model based Dual Heuristic Programming
DQN	=	Deep Q-Network
TD	=	temporal difference
DASMAT	=	Delft University Aircraft Simulation Model and Analysis Tool
TRPO	=	Trust Region Policy Optimization
DRL	=	Deep Reinforcement Learning
ADP	=	Approximate Dynamic Programming
ACD	=	Adaptive Critic Designs
HDP	=	Heuristic Dynamic Programming
SHERPA	=	Safety Handling Exploration with Risk Perception Algorithm
PID controller	=	Proportional Integral Derivative controller
TPM	=	Transient Performance Measurement
SSS	=	Safe State Space

List of Algorithms

1 Introduction

This chapter gives an introduction to the thesis report: ”Safe Online Continuous Reinforcement Learning for Flight Control”. The report is written at the Control & Simulation department of the Faculty of Aerospace Engineering of Delft University of Technology. First the motivation for the research is presented, this is followed by an overview of the research objectives, research questions and the scope of the research in section 1.2. Lastly an outline of the report is given.

1.1. Motivation

Aviation is widely regarded as the safest means of transport [63]. Continuous advancements from the aviation industry alongside with strict international rules make that the number of incidents per year are still declining[1], whilst the total number of flights are increasing[2]. The introduction of Flight Control Systems (FCS) has enhanced this safety by adding closed loop stability, putting boundaries on pilots inputs and reducing the pilots workload [17]. FCSs have been designed using linear control theory for many years, with satisfactory results. Nonetheless, linear control theory suffers from performance degradation due to non-linearities, uncertainties in the model, and faults or damage taken by the aircraft in reality. Furthermore the design of FCSs using linear control theory is costly due to required gain scheduling because of the large range of dynamics in the operating range of the aircraft. Linear control theory is also not optimal for coupled systems such as aircraft [2]. This performance degradation can also lead to safety issues when damage occurs. ICAO found that in the period of 2006-2010 Loss of Control was number one cause for fatalities in aviation [33]. Although not every incident was a consequence of linear control theory being used, the use of adaptive flight control can improve the safety of aircraft in unexpected circumstances.

In order to combat the performance degradation a lot of research has been performed and proven. Doyle, Lenz, and Packard [22] have shown that H_∞ loop shaping in combination with μ-synthesis allows for the shaping of the controller via a mathematical optimization function such that the robustness and performances are balanced even when uncertainties are present. However this method is often complex to apply and is still based on linear models. Furthermore Kulcsár [43] has shown how a Linear Quadratic Regulator (LQR) can be used to control an aircraft. LQR make use of a quadratic cost function to find the optimal solution that minimizes undesired deviations. Again this is a linear method and therefore is less effective as the system becomes more complex and non-linear.

Next Non-linear Dynamic Inversion (NDI) has shown to be able to cope with non-linearities as it linearizes the system with an inner loop [55], but lacks robustness to uncertainties and sensor noise. [60, 47] have presented Incremental Non-linear Dynamic Inversion (INDI) is presented which is more robust and can handle uncertainties and sensor noise. All the above methods have been used in actual flight [6, 25] and showed improved performance over classical linear control methods, they can deal with uncertainties. In particular NDI and INDI are interesting as they can also cope with non linearties, NDI and INDI also have the ability to adapt to changing aircraft behavior due to faults or damage when

[1]`http://www.baaa-acro.com/statistics/crashs-rate-per-year`

[2]`https://www.statista.com/statistics/564769/airline-industry-number-of-flights/`

assisted by online model identification. However implementation of the above mentioned methods can be mathematically complex and its performance dependent on the sensor accuracy.

The advancement of reinforcement learning in recent years along with its highly adaptive character made that is has also found its way into the field of control theory and thus also the field of adaptive flight control. In the field of computer science state-of-the-art reinforcement learning algorithms using neural networks combined with an increase in computing power have resulted in high performance on simple and complex tasks. In 2015 the DDPG algorithm [44] showed how reinforcement learning can be used on continuous tasks ranging from simple pendulum swings to more complex locomotion tasks. Furthermore reinforcement learning has also shown to be effective in adapting to changing circumstances [7]. In this research agents were trained for a particular task, but when put in a different environment, they could use their learned skills in an innovative way in order to survive.

In reinforcement learning an agent must learn from its own experiences through trial and error. It does not have any information with regards to the system it is controlling or what states can be expected depending on the actions taken. By means of a reward signal it is reinforced and receives feedback on its decisions. Once fully trained, a reinforcement learning agent is able select the actions resulting in the highest rewards and achieves optimal performance. When system dynamics change, the agent will see its reward signal change accordingly and adapts its policy. Many exploring trials may be required before an optimal policy is found. Additionally, exploration can be dangerous as fatal states can be encountered from which further learning is impossible. In many instances problems can only be solved using reinforcement learning when specific hyper parameters are chosen. As an aircraft faces many different tasks this is undesirable and optimally the controller is robust to these hyper parameter settings.

Reinforcement learning methods such as Heuristic Programming, Dual Heuristic Programming and Incremental Dual Heuristic Programming [76], recently applied on aerospace systems, have shown that high performance can be acquired whilst also being robust to unforeseen changes such as faults and damage to the aircraft. The control law will automatically adapt itself. However, no guarantees about safety can be given as the controller needs to explore in order to learn. This concern for safety during exploration is not just evident in the aerospace industry but a general drawback of using reinforcement learning as evident from e.g. [48, 41, 23, 21].

In this research an investigation into state-of-the art safe online reinforcement learning methods is performed and applied into the field of flight control. The model free character of reinforcement learning in combination with the demonstrated robustness to aircraft damages or faults offers many advantages that could be of use in the field of adaptive flight control. However, before these methods can be applied on a real aircraft, guarantees about safety have to be given [34]. The main goal of the research is to develop a safe online reinforcement learning flight controller that retains the adaptiveness exhibited in previous research [76] whilst also observing predefined safety limits. The controller should be robust to both changes in the environment as well as to different learning settings.

1.2. Research Objective and Questions

A complete solution for safe online reinforcement learning in flight control is still far from completion. In order to guide this research, a research objective is defined. Furthermore, research questions are used to structure the study and to create a well supported conclusion. Next, to limit the study and to create an achievable goal in the time available, a scope is provided.

The objective of this research is to *increase the safety of aerospace systems through online reinforcement learning by developing a state-of-the-art safe reinforcement learning method and by comparing its performance to existing online reinforcement learning methods.*

The following questions are used to structure the research. Each question provides more knowledge and experience of a specific sub-problem such that, after all question have been answered, the research objective is met.

1 What is the state-of-the-art in the field of online reinforcement learning?

- What training methods are used for online reinforcement learning
- What is the state-of-the-art in the field of safe online reinforcement learning?
- What is the state-of-the-art in the field of safe online reinforcement learning in flight control?

2 How is a trade-off performed between safety and performance in reinforcement learning?

- How is safety compared/measured?
- How is performance compared/measured?

3 How is safety guaranteed when using reinforcement learning in flight control?

- How is safety defined in flight control?
- What (initial) information needs to be available to the system to ensure safety?
- What methods are available to ensure safety in initialization and exploration?

4 How does the performance of the selected safe online continuous reinforcement learning method compare to existing continuous reinforcement learning controllers, such as (i)ADP?

- How does safety compare?
- How adaptive is the safe reinforcement learning method compared to existing reinforcement learning controllers?

Question 1 will help provide the necessary knowledge and state-of-the-art methods that are required to fulfill the research objective. Next, answering question 2 provides an insight in safety from the perspective of reinforcement learning. In order to successfully apply the knowledge on flight control problems question 3 is used. Additionally question 3 also gives a view on safety based on current flight control standards. Lastly, a comparison is made with current state of the art methods by answering question 4.

From the objective and the research questions the scope of the research can be limited to continuous online reinforcement learning on a flight control problem. The focus is mainly on safety and robustness of the algorithm and secondly on performance. A comparison between the proposed methods is performed using a tracking task, whilst the information about the aircraft dynamics is unknown beforehand. As no aircraft dynamics are known by the reinforcement learning controller, initialization of the simulations will not take place near failure states. Furthermore, known aircraft state limits are also given to the algorithm. Lastly, as this research is a proof of concept, only longitudinal control is studied.

1.3. Outline

This report consist of 4 sections. First in Part I the research is presented in the formal of a scientific paper. This can be read as a standalone document and provides the reader with a full understanding of the research carried out. Furthermore, relevant conclusions and suggestions for further research are also included. In Part II the basis of the research is given. chapter 2 presents a literature overview of both the basics and state-of-the-art of reinforcement learning and safe reinforcement learning. Also the topic of safety in flight control is touched upon. Next chapter 3 presents the preliminary analysis performed, which presents a proof of concept using tabular q-learning. The proof of concept led to the framework studied in this research. Lastly, Part III shows additional results not presented in the paper. This includes examples of individual runs on the height trajectory task in chapter 4, a discussion of the accuracy and usability of the predicted states in chapter 5, and a discussion regarding the provided learning impulse in chapter 6. Next an extensive conclusion answering all research questions is provided in chapter 7 and finally recommendations for future research are given in chapter 8.

I

Scientific Article

Safe Online Continuous Reinforcement Learning for Flight Control

Rick R. Feith[*]
Delft University of Technology, P.o. Box 5058, 2600GB Delft, Nederland

Online continuous reinforcement learning has shown promising result in flight control achieving near optimal control within seconds and the capability to adapt to sudden changes in the system. However no guarantees about safety and performance can be given, needed for use in general aviation. Furthermore performance is often dependent on the precise tuning of the hyper parameters inside the system. These issues are not solely faced in aviation, but encountered in every safety-critical sector. As an initial step in providing guarantees about safety and performance, this paper presents Safe Incremental Dual Heuristic Programming (SIDHP). SIDHP combines the fast learning speed of Incremental Dual Heuristic Programming (IDHP) with a safety layer, able to keep the aircraft within a predetermined safe flight envelope during training. SIDHP is demonstrated and compared to IDHP using a high fidelity nonlinear flight simulation of a Cessna Citation II in three separate experiments. SIDHP is more robust with respect to changing hyper parameters compared to IDHP allowing for a more general application. Furthermore it is shown that SIDHP does not reduce convergence speed and is still able to cope with sudden changes within the system.

Nomenclature

$\boldsymbol{s}$	=	state vector
$\boldsymbol{a}_t, \boldsymbol{a}_p, \boldsymbol{a}_c, \boldsymbol{a}_s$	=	action vector, proposed action vector, corrective action vector, safe action vector
L^a, L^C	=	Actor and Critic Loss
r	=	reward
t	=	current timestep
$\boldsymbol{W}^C, \boldsymbol{W}^A$	=	critic neural network weights, actor neural network weights
η^C, η^A, η^s	=	critic learning rate, actor learning rate, safe learning rate multiplier
$\boldsymbol{F}, \hat{\boldsymbol{F}}$	=	state matrix, estimated state matrix
$\boldsymbol{G}, \hat{\boldsymbol{F}}$	=	input matrix, estimated input matrix
γ	=	discount factor
$\boldsymbol{e}$	=	error vector
$v(\boldsymbol{s})$	=	value function
$\lambda(\boldsymbol{s})$	=	partial derivative of the value function with respect to each state
$\pi(\boldsymbol{s})$	=	policy function
Θ	=	RLS weights
Λ	=	RLS covariance matrix
$\boldsymbol{X}$	=	RLS measurement matrix
$\boldsymbol{\epsilon}$	=	RLS prediction error
k	=	RLS forgetting factor
n, q, α, θ, V, H	=	load factor, pitch rate, angle of attack, pitch attitude, Velocity, Height
δ_e	=	elevator deflection

[*]MSc Student, Faculty of Aerospace Engineering, Control and Simulation Department, Delft University of Technology.

I. Introduction

Aviation is widely regarded as the safest means of transport [1]. Continuous advancements from the aviation industry alongside with strict international rules make that the number of incidents per year are still declining*, whilst the total number of flights are increasing†. The introduction of Flight Control Systems (FCS) has enhanced this safety by adding closed loop stability, putting boundaries on pilots inputs and reducing the pilots workload [2]. FCSs have been designed using linear control theory for many years, with satisfactory results. Nonetheless, linear control theory suffers from performance degradation due to non-linearities, uncertainties in the model, and faults or damage taken by the aircraft in reality. Furthermore the design of FCSs using linear control theory is costly due to required gain scheduling because of the range of dynamics in the operating range of the aircraft and the need for an accurate model which might not be readily available [3]. The performance degradation can also lead to safety issues when damage occurs. With the rise of autonomous systems this need for adaptive control and safety becomes even more apparent.

In order to combat the performance degradation a lot of research has been performed and proven on aircraft. For example Doyle, Lenz, and Packard [4] have shown H_∞ loop shaping in combination with μ-synthesis, whilst Kulcsár [5] has shown how a Linear Quadratic Regulator (LQR). Both allows for the shaping of the controller such that the robustness and performances are balanced. However both are mathematically complex linear methods and therefore less effective as the system becomes more complex and non-linear.

Next Non-linear Dynamic Inversion (NDI) has shown to be able to cope with non-linearities as it linearizes the system with an inner loop [6], but lacks robustness to uncertainties and sensor noise. In [7] Incremental Non-linear Dynamic Inversion (INDI) is presented which is more robust and can handle uncertainties and sensor noise. These methods have been used in actual flight [8, 9] and showed improved performance over classical linear control methods. In particular NDI and INDI are interesting as they can cope with uncertainties, non-linearities and have the ability to adapt to changing aircraft behavior due to faults or damage. However implementation of the above mentioned methods can be mathematically complex and its performance dependent on the sensor accuracy.

The advancement of reinforcement learning in recent years made that it has found its way into the field of adaptive flight control. In the field of computer science state-of-the-art reinforcement learning algorithms using neural networks have resulted in high performance on simple and complex tasks. The DDPG algorithm [10] showed how reinforcement learning can be used on continuous tasks ranging from simple pendulum swings to more complex locomotion tasks. Furthermore, reinforcement learning has also shown to be able to adapt to changing circumstances [11].

In fundamental reinforcement learning an agent must learn from its own experiences through trial and error. It does not have any information with regards to the system it is controlling. By means of a reward signal it is reinforced and receives feedback on its decisions. Once trained, a reinforcement learning agent is able to select the actions resulting in the highest rewards and achieves optimal performance. When system dynamics change, the agent will see its reward signal change accordingly and adapts its policy. Many exploring trials may be required before an optimal policy is found. Additionally, exploration can be dangerous as fatal states can be encountered from which further learning is impossible. In many cases convergence is only achieved with specific hyper parameter settings.

Reinforcement learning methods such as Heuristic Dynamic Programming (HDP), Dual Heuristic Dynamic Programming (DHP) and Incremental Dual Heuristic Dynamic Programming (IDHP) [12], recently applied on aerospace systems, have shown that high online performance can be acquired whilst also being robust to unforeseen changes such as faults and damage to the aircraft. However, no guarantees about safety can be given as the controller needs to explore in order to learn. This concern for safety during exploration is not just evident in the aerospace industry but a general drawback of using reinforcement learning as evident from e.g. [13–16].

The contribution of this research is a Safe IDHP (SIDHP) reinforcement learning flight controller. The model free character of reinforcement learning in combination with the demonstrated robustness to aircraft damages or faults offers many advantages that could be of use in the field of adaptive flight control. However, before these methods can be applied on a real aircraft, guarantees about safety have to be given [17]. SIDHP can learn a near optimal policy online whilst a safety layer keeps the aircraft from entering unsafe states. This is done by predicting the next states, using the chosen action, and replacing the action if the predicted state is deemed unsafe. The entire framework is demonstrated on a high fidelity non-linear model of a Cessna Citation II. SIDHP shows robustness to both changes in the environment as well as to different learning settings.

The paper is structured as follows. First, in section II, previous research used is summarized. Next, in section III the concept of SIDHP is explained. This is followed by the experimental setup in section IV. section V presents the results. Lastly, section VI provides the conclusion and recommendations for future work.

*http://www.baaa-acro.com/statistics/crashs-rate-per-year

†https://www.statista.com/statistics/564769/airline-industry-number-of-flights/

II. Methodology

In this section a short motivation for the choice of IDHP is given followed by an explanation of IDHP. A background into safe reinforcement learning is provided after which a novel safe approach for aircraft control is presented.

A. Incremental Dual Heuristic Programming

In reinforcement learning an agent learns to control an environment through trial and error. The only information received is the current state and the accompanying reward, the output consequently is the action taken. This is also visualized in Figure 3. The agents objective is to maximize the total cumulative reward, or return G_t, which corresponds to finding a near optimal control policy.

When using a reinforcement learning controller in order to control an aircraft in an online fashion, the controller needs to learn how to control the aircraft before the aircraft is out of control. Thus a method is required that converges quickly. Incremental Dual Heuristic Dynamic Programming (IDHP), as presented in [12], is chosen as a starting point. This method proved to be capable of learning online. IDHP uses an actor neural network, a critic neural network, and an incremental model to model the environment. Knowledge about the environment is more efficiently gained through the use of the incremental model, such that online learning is possible even for non-linear environments. As an added benefit of this method the incremental model generated by this method can also be used to predict future states, which are used by SIDHP as shown in subsection II.B.

The function approximators for the critic and actor are small fully connected neural networks with 1 hidden layer. These are able to approximate any non-linear function in a compact space, are differentiable, and already widely used in reinforcement learning and intelligent control systems[11, 14, 18]. Furthermore it should be noted that although in classical reinforcement learning the reward is determined by the environment, in the case of IDHP the reward function is designed and known such that it is differentiable.

First the incremental model is derived, after which the critic and actor networks are described along with their update rules. The definitions and mathematical formulas used are taken from [12]. [19] or [20] as indicated. Vectors are written in bold.

1. Incremental Model

In fundamental reinforcement learning no knowledge of the system is known beforehand and all should be learned by the agent through trail and error. In order to speed up this process IDHP also learns a incremental model of the system which is then used to more accurately provide gradient updates to the critic and actor. This incremental model can be seen as a first-order Taylor expansion of the discrete non-linear plant at a point $[\boldsymbol{s}_t, \boldsymbol{a}_t]$ as shown in Equation 1 [19]. Here $\boldsymbol{F}$ and $\boldsymbol{G}$ are also known as the state and input matrix respectively and can be seen as the state-transition functions.

$$\boldsymbol{s}_t \approx \boldsymbol{s}_{t-1} + \frac{\partial f(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})}{\partial \boldsymbol{s}_{t-1}} [\boldsymbol{s}_{t-1} - \boldsymbol{s}_{t-2}] + \frac{\partial f(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})}{\partial \boldsymbol{a}_{t-1}} [\boldsymbol{a}_{t-1} - \boldsymbol{a}_{t-2}] = \boldsymbol{s}_{t-1} + \boldsymbol{F}\Delta \boldsymbol{s}_{t-1} + \boldsymbol{G}\Delta \boldsymbol{a}_{t-1} \tag{1}$$

In order to estimate the matrices $\boldsymbol{F}$ and $\boldsymbol{G}$, Recursive Least Squares (RLS) is used. In addition to $\boldsymbol{F}$ and $\boldsymbol{G}$ RLS requires a forgetting factor, k, which weights the measurements exponentially, reducing the weight of older measurements. Furthermore a covariance matrix, Λ, is used to indicate the confidence and update rate of the model.

RLS first compiles $\boldsymbol{F}$ and $\boldsymbol{G}$ into $\boldsymbol{\Theta}$, and $\Delta \boldsymbol{s}$ and $\Delta \boldsymbol{a}$ into $\boldsymbol{X}$ as seen in Equation 2 and 3. Next, an estimated next change in state, $\Delta \hat{\boldsymbol{s}}$, is calculated according to Equation 4. The error between the actual change in state, $\Delta \boldsymbol{s}$, and the estimated change in state, $\Delta \hat{\boldsymbol{s}}$, can then be found using Equation 5. From here using Equation 6 and Equation 7 the RLS model is updated. Equation 2, 3, 4, 5, 6 and 7 are taken from [12].

$$\hat{\boldsymbol{\Theta}}_{t-1} = \begin{bmatrix} \hat{\boldsymbol{F}}_{t-1}^T & \hat{\boldsymbol{G}}_{t-1}^T \end{bmatrix}^T \tag{2}$$

$$\boldsymbol{X}_t = [\Delta \boldsymbol{s}_{t-1} \quad \Delta \boldsymbol{a}_{t-1}]^t \tag{3}$$

$$\Delta \hat{\boldsymbol{s}}_t^T = \boldsymbol{X}_{t-1}^T \hat{\boldsymbol{\Theta}}_{t-1} \tag{4}$$

$$\boldsymbol{\epsilon}_t = \Delta \boldsymbol{s}_t^T - \Delta \hat{\boldsymbol{s}}_t^T \tag{5}$$

$$\hat{\boldsymbol{\Theta}}_t = \hat{\boldsymbol{\Theta}}_{t-1} + \frac{\boldsymbol{\Lambda}_{t-1}\boldsymbol{X}_{t-1}}{k + \boldsymbol{X}_t^T \boldsymbol{\Lambda}_{t-1} \boldsymbol{X}_t} \boldsymbol{\epsilon}_t \tag{6}$$

$$\boldsymbol{\Lambda}_t = \frac{1}{k}\left[\boldsymbol{\Lambda}_{t-1} - \frac{\boldsymbol{\Lambda}_{t-1}\boldsymbol{X}_t\boldsymbol{X}_t^T\boldsymbol{\Lambda}_{t-1}}{k + \boldsymbol{X}_t^T\boldsymbol{\Lambda}_{t-1}\boldsymbol{X}_t}\right] \tag{7}$$

2. *Critic network*

The role of the critic is to provide feedback to the actor by learning to estimate the partial derivative of the value function,$v(\boldsymbol{s})$, with respect to each state element, indicated by $\lambda(\boldsymbol{s})$. This partial derivative is then used by the actor to optimize for the maximum return. As explained IDHP requires a reward function that is differentiably. For this study the reward function is defined as the negative of the squared error between the state and the reference state. This can be seen in Equation 8, Equation 9 shows the property of the value function.

$$r_t = -(\boldsymbol{s}_t - \boldsymbol{s}_t^{ref})^2 \tag{8}$$

$$v_t = r_t + v_{t-1} \tag{9}$$

The loss function used to optimize the critic is defined as the mean squared error, as seen in Equation 10 [19, 20] and needs to be minimized. The error, $\boldsymbol{e}$, used can be found in Equation 11 [19, 20] and can interpreted as the partial derivative of the temporal difference error with respect to the state vector. Here the need for a differentiable reward function becomes also clear. A discount factor, γ, is used to discount future rewards, $\hat{\lambda}$ is the output of the critic network.

$$L_t^C = \frac{1}{2}\boldsymbol{e}_t\boldsymbol{e}_t^T \tag{10}$$

$$\begin{aligned} \boldsymbol{e}_t &= \frac{\partial\left[v(\boldsymbol{s}_{t-1}) - (r_t + \gamma v(\boldsymbol{s}_t))\right]}{\partial \boldsymbol{s}_{t-1}} \\ &= \hat{\lambda}(\boldsymbol{s}_{t-1}, \boldsymbol{w}_t^C) - \left[\frac{\partial r(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t} + \gamma\hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{w}_t^C)\right]\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_{t-1}} \end{aligned} \tag{11}$$

Using the learned incremental model as shown in Equation 12 [19, 20] gives us the term $\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_{t-1}}$. Here $\frac{\partial \hat{\pi}(\boldsymbol{s}_{t-1}, \boldsymbol{w}_{t-1}^A)}{\partial \boldsymbol{s}_{t-1}}$ is the current policy by the actor. As the actor is also a neural network it is guaranteed to be differentiable.

$$\begin{aligned} \frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_{t-1}} &= \frac{\partial f(\boldsymbol{s}_t, \boldsymbol{a}_t)}{\partial \boldsymbol{s}_t} + \frac{\partial f(\boldsymbol{s}_t, \boldsymbol{a}_t)}{\partial \boldsymbol{a}_t}\frac{\partial \boldsymbol{a}_t}{\partial \boldsymbol{s}_t} \\ &\approx \hat{\boldsymbol{F}}_t + \hat{\boldsymbol{G}}_t \frac{\partial \hat{\pi}(\boldsymbol{s}_{t-1}, \boldsymbol{w}_{t-1}^A)}{\partial \boldsymbol{s}_{t-1}} \end{aligned} \tag{12}$$

The total gradient used for the network can then be found using Equation 13 [19, 20], with $\frac{\partial \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{w}_t^C)}{\partial \boldsymbol{w}_t^C}$ being provided by the neural networks internal learning framework. This gradient is then applied according to Equation 14 [19, 20] where η represents the learning rate. The critic is updated at each timestep.

$$\begin{aligned} \frac{\partial L_t}{\partial \boldsymbol{w}_t^C} &= \frac{\partial L_t^C}{\partial \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{w}_t^C)}\frac{\partial \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{w}_t^C)}{\partial \boldsymbol{w}_t^C} = \boldsymbol{e}_t \frac{\partial \hat{\lambda}(\boldsymbol{s}_t, \boldsymbol{w}_t^C)}{\partial \boldsymbol{w}_t^C} \\ &= \left[\hat{\lambda}(\boldsymbol{s}_{t-1}, \boldsymbol{w}_t^C) - \left(\frac{\partial r_t}{\boldsymbol{s}_t} + \gamma\lambda(\boldsymbol{s}_t)\right)\left(\hat{\boldsymbol{F}}_t + \hat{\boldsymbol{G}}_t \frac{\partial \hat{\pi}(\boldsymbol{s}_{t-1}, \boldsymbol{w}_{t-1}^A)}{\partial \boldsymbol{s}_{t-1}}\right)\right]\frac{\partial \lambda(\boldsymbol{s}_t)}{\partial \boldsymbol{w}_t^C} \end{aligned} \tag{13}$$

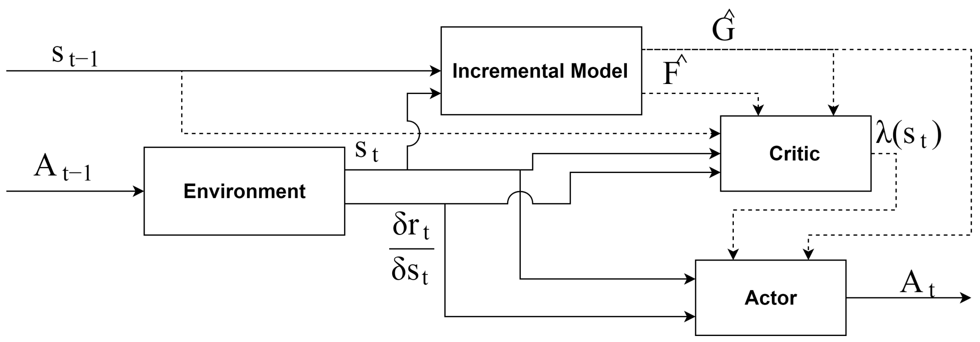

Fig. 1 Overview of informational flow when IDHP is used. Dotted lines represent feedback loops used for learning.

$$w_{t+1}^C = w_t^C - \eta^C \frac{\partial L_t}{\partial w_t^C} \tag{14}$$

3. Actor network

The goal of the actor network is to find a policy that yields the maximum reward, or $\pi(s_t) = \arg\max_{a_t} v(s_t)$. As such the the loss function used to optimize is defined in Equation 15 [19, 20].

$$L_t^A = -v(s_t) = -\left[r(s_t) + \gamma v(s_t)\right] \tag{15}$$

The learning rule used to update the actor can thus be derived and this is shown in Equation 16 [19, 20]. The need of the learned incremental model is again visible in $\hat{G}_t$. Additionally it can also be seen how the critic network, which outputs $\lambda(\hat{s_t})$, is used and affects the performance of the agent. Lastly the weight updates are performed according to Equation 17 [19, 20]. The actor is also updated during each timestep.

$$\begin{aligned}
\frac{\partial L_t}{\partial w^A} &= -\frac{\partial\left[r_t + \gamma v(s_t)\right]}{\partial w_t^A} \\
&= -\left[\frac{\partial r_t}{\partial s_t} + \gamma \frac{\partial v(s_t)}{\partial s_t}\right] \frac{\partial s_t}{\partial a_{t-1}} \frac{a_{t-1}}{w_t^A} \\
&\approx -\left[\frac{\partial r_t}{s_t} + \gamma \hat{\lambda}(s_t)\right] \hat{G}_t \frac{\partial a_t}{\partial w^A}
\end{aligned} \tag{16}$$

$$w_{t+1}^A = w_t^A - \eta^A \frac{\partial L_t}{\partial w^A} \tag{17}$$

A complete overview of the informational flow used in IDHP can be found in Figure 1.

B. Safe Reinforcement Learning

Finding a near optimal control policy can take many trials as in theory all state-action pairs will have to be visited to find the optimal policy. If not enough state-action pairs are explored the optimal policy may not be found, spending too much time exploring is also not desirable as this time could be used to gather rewards instead. This problem is known as the exploration-exploitation dilemma[13]. Additionally, not all state-action pairs are safe or lead to safe states which makes exploring dangerous. In practice this often means that a reinforcement learning controller only works with a very specific initialization and learning rate and even then no guarantees about safety can be given. During computer simulations the worst that can happen is a failed trial run. In real life exploring the entire state-space could prove dangerous. An aircraft could end up in an unrecoverable stall or dive, or a robot could possibly damage itselves or its surroundings. Therefore a lot of research has already been performed on safe reinforcement learning. Here a short overview of safe reinforcement learning is provided.

In order to increase the safety of reinforcement learning during exploration one must realize reinforcement learning makes use of exploration in order to obtain knowledge about the environment. If exploration is therefore limited somehow, external knowledge needs to be provided [15, 21].

A simple and widely adopted approach is to simply penalize unwanted states. As such the agent will learn to avoid these [22]. In this case extra knowledge is added to the reward function. However, the agent will still have to experience these states before knowing that a penalty is inflicted and therefore for aerospace control this is not a desirable solution. Additionally, a risk-sensitivity approach can be taken by the agent in which case not only reward, but also the risk is taken into account in the decision process [15]. Combining this with a backup trajectory to a known safe state as shown in [13] has yielded promising results. However, these method often require a conservative but safe controller as a starting point. This controller is not always known.

Furthermore "learning from demonstration", in which case the agent tries to emulate a (human) controller, has also been studied extensively [15, 22, 23]. Here knowledge is added in the form of safe examples. Another approach, found in [24], uses reinforcement learning to tune a safe, but not optimal controller, such that safety is guaranteed but performance increases. Again, for this method to work, an initial controller has to be available. Next, [14] uses reinforcement learning in combination with robust control theory. The robust controller maintains stability and reinforcement learning increases performance. For this approach an uncertain model of the environment needs to be available. As a last example, in [16] a safety layer is used in order to constrain the output of the agent to actions that are known to keep the environment in a safe state. Here an offline training phase is used to generate the safety layer such that it knows which actions and states are safe.

It can be seen that each method requires some form of external knowledge to work. For the problem this study tries to solve, namely online safe exploration without prior knowledge of the environment or controller, however none seem a fit solution. Therefore this study presents a novel adaptation of IDHP, Safe IDHP, increasing the safety in reinforcement learning problems without prior knowledge of the controller or environment.

III. Safe Incremental Dual Heuristic Dynamic Programming

In this section the application of Safe Incremental Dual Heuristic Dynamic Programming (SIDHP) is presented. The specific problem of safe online continuous reinforcement learning for flight control requires the controller to learn online, preferably without prior knowledge, but to also learn safe. As such SIDHP is developed based on IDHP [12] and the concept of a safety layer as presented in [16].

A. Safety in Flight Control

In flight control safety is expressed in the stability of the aircraft, and the safe flight envelope the aircraft has to adhere to. In reinforcement learning these terms would coincide with "convergence" and adhering to the "safe states." A stable aircraft will contribute to safety as it will return to its initial state after a disturbance such as a windgust. Furthermore, to increase performance an aircraft is often tuned to be slightly underdamped [25]. For reinforcement learning stability is difficult to express, as the controller is constantly changing. This also creates issues during certification of the controller. The current tests and metrics used are unable to certify adaptive controllers, as expressed by [26] and [27]. This however is outside the scope of this research.

The safe flight envelope of an aircraft indicates in what states the aircraft can fly as directed by the design limits. As opposed to stability, which can be improved through tuning of the controllers, the design limits can not be altered. For example, aerodynamic constraints can lead to a maximum angle of attack, and structural constraints have an influence on the maximum load factor. Furthermore the current state of the aircraft, such as velocity and aircraft weight affect the safe flight envelope. Modern aircraft often have a flight envelope protection system to increase the safety during operation [2]. Additionally during flight, altitude and speed limits can be given by the local authorities. Identifying these limits is a complex procedure and not non-trivial, but outside the scope of the research. The safe flight envelope is directly correlated with the safe states the aircraft can explore using a reinforcement learning controller. For the specific problem of safe online continuous reinforcement learning for flight control it is thus known in advance what the safe states space (SSS) [13] is under normal operation. This knowledge can thus be provided in advance.

B. Safety Layer

The addition of a safety layer as presented in [16] ensures that actions leading to an unsafe state are replaced by an action leading back to a safe state much like a flight envelope protection system for PID control. A comparison

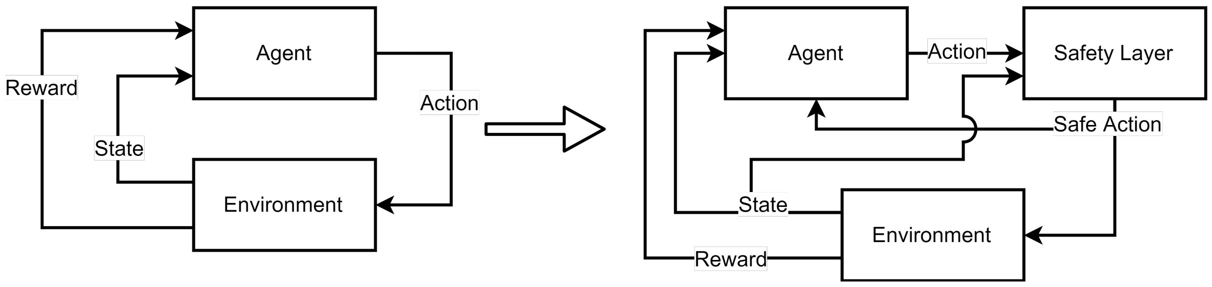

Fig. 2 Reinforcement learning framework transition by the addition of a safety layer.

with respect to fundamental reinforcement learning can be found in Figure 2. The same concept is applied on IDHP. In [16] first an offline learning phase is performed to learn the safe states and an internal constraint model. This is undesirable as a solution for online learning is required. With the assumption that the safe states are known in advance through the safe flight envelope this offline learning phase is not required. Additionally, the incremental model included in IDHP provides a basis for predicting future states and can be used to determine if the safety layer should replace the current action. Benefits of using an incremental linear model instead of using a second non-linear model trained trough reinforcement learning have been expounded upon in [12], the general motivation is the speed of convergence for linear models.

The safety layer performs four steps to ensure the aircraft stays within its safe flight envelope. First, a prediction of the future state is calculated using the learned incremental model contained in IDHP using Equation 18. By repeating the calculation $\boldsymbol{s}_{t+n}$ can be found, where n indicates the number of future time steps to be predicted. Increasing this number also increases the uncertainty of the result since a prediction of a prediction is made. Furthermore, the algorithm makes the prediction assuming the same action is applied at each timestep, this can differ from reality.

$$\hat{\boldsymbol{s}}_{t+1} \approx \boldsymbol{s}_t + \hat{\boldsymbol{F}}_t \Delta \boldsymbol{s}_t + \hat{\boldsymbol{G}}_t \Delta \boldsymbol{a}_t \tag{18}$$

Secondly, a check is performed whether the predicted state vector is within the safe flight envelope or not. If the predicted state vector is within the boundaries the learning algorithm continues without intervention. If the predicted state vector is outside of a boundary a new action is proposed that brings the aircraft further back into the safe state space. The new action is determined by looking at the sign of $\hat{\boldsymbol{G}}_t$ of the corresponding state that exceeds the boundaries. In order to keep the process relatively smooth the previous action $\boldsymbol{a}_{t-1}$ is taken as starting point and from there the safety layer adjusts the action according to Equation 19 where $\boldsymbol{a}_c$ is a constant corrective parameter and $\boldsymbol{a}_s$ represent the safe action. This smoothness is required to ensure that the learned models and controllers have time to adapt and no discontinuities are encountered.

$$\boldsymbol{a}_s = \boldsymbol{a}_{t-1} + \text{signum}(\hat{\boldsymbol{G}}_t)\boldsymbol{a}_c \tag{19}$$

Lastly, in order for the agent to learn from this action the learning process is now altered for the actor. Instead of using Equation 15 and 16 to find $\frac{\partial L_t}{\partial \boldsymbol{w}^A}$ Equation 20 and 21 are used. Here $\boldsymbol{a}_t$ indicates the proposed action by the actor, η^s is a learning rate multiplier to increase the impact of the safety layer. Basically the actor learns to emulate the safety layer when it is activated. With this step the actor is provided information with regards to the safe states and is able to learn the bounds of the safe flight envelope. This can be compared to "learning from demonstration" [23].

$$L_t^A = -\eta^s(\boldsymbol{a}_s - \boldsymbol{a}_t)^2 \tag{20}$$

$$\frac{\partial L_t}{\partial \boldsymbol{w}^A} = -2\eta^s(\boldsymbol{a}_s - \boldsymbol{a}_t) \tag{21}$$

$$\tag{22}$$

To further smoothen the process, the safety layer is only active if it was not active the previous timestep. This allows for more accurate critic updates as no discontinuties in the states are encountered. If the safety layer were to bring the aircraft into an entire different part of the state space in one timestep, the current critic and actor might not be valid

Algorithm 1 Overview of Safety Layer

```
Require: IDHP-agent, Safe State Space, simulation environment, $\boldsymbol{a}_c$
  $\boldsymbol{a}_t, \hat{\boldsymbol{F}}_t, \hat{\boldsymbol{G}}_t \leftarrow$IDHP-agent
  $\boldsymbol{s}_t \leftarrow$environment
  for $i$ in $n$ do
    $\boldsymbol{s}_{t+i+1} \leftarrow \boldsymbol{s}_{t+i} + \hat{\boldsymbol{F}}_t \Delta \boldsymbol{s}_{t+i} + \hat{\boldsymbol{G}}_t \Delta \boldsymbol{a}_t$
  end for
  if $\boldsymbol{s}_{t+n} \in SSS$ then
    $\frac{\partial L_t}{\partial w^A} \leftarrow -\left[\frac{\partial r_t}{s_t} + \gamma \hat{\lambda}(\boldsymbol{s}_t)\right] \hat{\boldsymbol{G}}_t \frac{\partial a_t}{\partial w^A}$
  else
    $\boldsymbol{a}_s \leftarrow \boldsymbol{a}_{t-1} + \text{signum}(\hat{\boldsymbol{G}}_t)\boldsymbol{a}_c$
    $\frac{\partial L_t}{\partial w^A} \leftarrow -2\eta^s(\boldsymbol{a}_s - \boldsymbol{a}_t)$
  end if
```

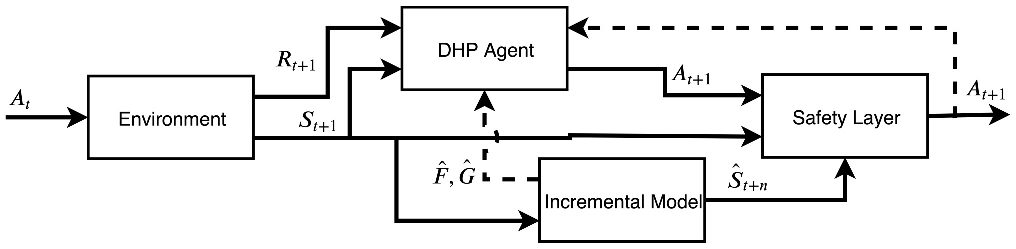

Fig. 3 Reinforcement learning framework transition by the addition of a safety layer.

there. A more gradual and smooth approach means they maintain their validity and performance. An overview of the informational flow is found in Figure 3, whilst Algorithm 1 contains the pseudocode.

In [16] a method is proposed to deal with multiple boundaries crossings at a single timestep. As SIDHP is applied to an aircraft control task an aviation approach is used. Upset recovery strategies have been researched extensively in the past and therefore the same strategy is applied here, boundaries are checked in the following order according to [28]:

1) Structural loading: the loadfactor n.
2) Angular accelerations: as only longitudinal tests are performed this consists of only the pitch rate q.
3) Stall limit: angle of attack α.
4) Airspeed: Velocity of the aircraft V.
5) Altitude: The height of the aircraft H.

As an example, an aircraft is flying lower than the reference and even below the minimum altitude. The controller therefore proposes an extreme pull-up manoeuvre. SIDHP predict the proposed action will result in a crossing of the maximum loadfactor and it detects that the aircraft is flying to low. Still it will replace the action with a less extreme action to avoid the structural limits as this has a higher priority.

IV. Experimental setup

In order to compare the performance and safety of IDHP and SIDHP three separate experiments are executed. The first experiment is a simple proof of concept tracking a pitch rate reference q^R, the second experiment compares safety and performance in nominal flight, by tracking a height trajectory H^R, the last experiment tests the controllers ability to deal with sudden changes, again tracking a pitch rate reference q^R. All experiments are performed using only longitudinal control.

A. Simulation Model

The experiments are performed using a high fidelity non-linear, six-degrees-of freedom model of a Cessna Citation 500, similar to the research aircraft used by the faculty of Aerospace Engineering of Delft University of Technology,

Table 1 I/O of Cessna Citation 500 model.

Input (action)	Input Limit	Output (state)
δ_e	$[-20.05, 14.90]$ °	q,θ,α,V,H,n, q^R or H^R

Fig. 4 PH-LAB research aircraft operated by Delft University of Technology and similar to the simulation model used [29].

developed with the Delft University Aircraft Simulation Model and Analysis Tool (DASMAT) [29, 30]. This model has also been used with success in previous research [19, 20]. For the experiments the model is initialized in a steady straight flight at an altitude of 2000 meter and a velocity of 90 m/s. Since only longitudinal motions are used, the ailerons and rudder are not used, only the elevator deflection is controlled by the agent. Unless stated otherwise the aircraft is in untrimmed condition and thus will slowly drift from its initial state. Time steps have a size of 0.2 seconds.

B. Controller Design

For experiment 1 and 3 a pitch rate reference,q^R is tracked and the reference signal outputted by the simulator can be used by the controller. For task 2 however, a height reference is given, which proved difficult to learn online. Other reinforcement learning methods such as Hierachical Reinforcement Learning [31] where one controller learns to map H^R to q^R and a second controller maps q^R to δ_e could provide an outcome. This is outside the scope of this research. Therefore an outerloop PID controller is used, as demonstrated in previous studies [19, 20]. The complete loop requires 2 PID controllers, as seen in Figure 5. For all experiments it is assumed that the measurement are clean.

C. Reinforcement Learning Controller parameters

For the IDHP and SIDHP controllers multiple (hyper-) parameters have to be configured. These are chosen based on literature and/or early experiments.

As inputs the controller receives q and q^R, it outputs δ_e. Additionaly the incremental model also receives α, γ,θ, V,

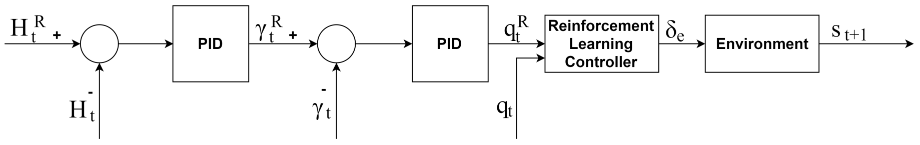

Fig. 5 Outer loop PID controller layout.

Table 2 Hyper parameters used by IDHP and SIDHP during the different experiments, values in bold are used in experiment 1 and 3, experiment 2 loops over all values.

Parameters	Values
η^{W_A},η^{W_C}	[0.1, 0.2, 0.5,**1**, 2,5], [0.2, 0.4, 1, **2**, 4, 10]
η^{B_A},η^{B_C}	[0, **0.0001**, 0.001], [0, **0.0002**, 0.002]
η^s	**10**
γ	**0.8**
W_0^A, W_0^C	$\mu = \mathbf{0}$, $\sigma = [0.001, \mathbf{0.01}, 0.1]$
B_0^A, B_0^C	**0**
$\hat{\boldsymbol{F}}_0, \hat{\boldsymbol{G}}_0, \Lambda_0$	$\mathbf{0}, \mathbf{0}, \boldsymbol{I}$
k	**1**
$\boldsymbol{a}_c$	**0.0035** rad
$t+n$	**10** steps or **0.2** sec

H, and n as input. These are required to predict future states. The choice could also be made to add all inputs to the controller, early tests showed that this did not increase either safety or performance, but did increase running time.

The neural network consists of 1 hidden layer with 1 node for both the actor and the critic. The hidden layer uses a hyperbolic tangent function whilst the output layer is linear. As only q and q^R are given as input to the network this proved most effective, decreasing converging times. An overview of all the hyperparameters that are used for the neural network, the incremental model and the safety layer can be found in Table 2. The bold parameters are the optimal parameters and used in experiment 1 and 3. Experiment 2 uses the complete range of values.

V. Results and Discussion

In order to compare the performance of SIDHP to IDHP three experiment are executed. Each experiment demonstrates a different ability of the algorithms.

A. Experiment 1: Tracking a Reference Pitch Rate

For the first experiment a reference pitch rate is generated. The algorithms have to successfully learn to track the reference. Additionally, a virtual bound is given on the maximum and minimum pitch rate, representing a safe flight envelope. These bounds get narrower as time progresses using exponential decay. The motivation being that the controllers should be able to more accurately follow the reference as time progresses and less space for exploration is required. This demonstrates the working principle of SIDHP.

The reference trajectory consist of a sinusoid with an amplitude of 5 ° and a frequency of 0.1 Hz. The run starts with random exitation to allow for faster converge. If the reference would move outside the bounds, the bound becomes the reference. In Figure 6 the results of the experiment can be seen for 100 runs using IDHP and 100 runs using SIDHP. Next in Figure 7 a randomly selected single run is shown and the different behaviour near the bound can clearly be seen. In Figure 8 the elevator deflection outputted by the respective agents can be found for a randomly selected single run.

For this particular task the overall tracking performance between IDHP and SIDHP is similar, as can be seen in Figure 6. The difference between IDHP and SIDHP becomes clear when looking at the behaviour near the bound, shown in Figure 7. It can be seen that IDHP tracks the reference well, but suffers from overshoot. As the reference follows the bound IDHP shows underdamped behaviour as it oscillates until the reference moves away from the bound again. In the case of SIDHP it can be seen that the algorithm predicts a crossing of the boundary and activates the safety layer. As a result the boundary is never crossed and after a short oscillation period SIDHP moves away slightly from the boundary. As the reference moves away from the bound SIDHP continues tracking the reference. That SIDHP moves away from the boundary can be explained by the learning impulse the safety layer gives to the network, according to Equation 21. As it learns to emulate the safety layer SIDHP learns to avoid the boundary. The same behaviour is seen in Figure 8, the safe action applied gives a learning impulse which reduces the maximum elevator deflection encountered.

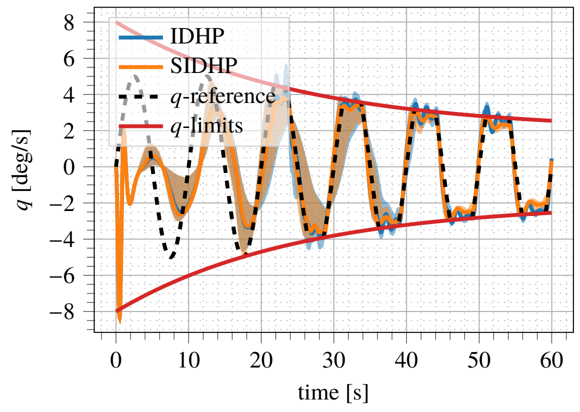

Fig. 6 Experiment 1: Tracking a reference pitch rate q. Solid lines represent the mean of 100 runs. The transparent area demonstrates the maximum and minimum values encountered.

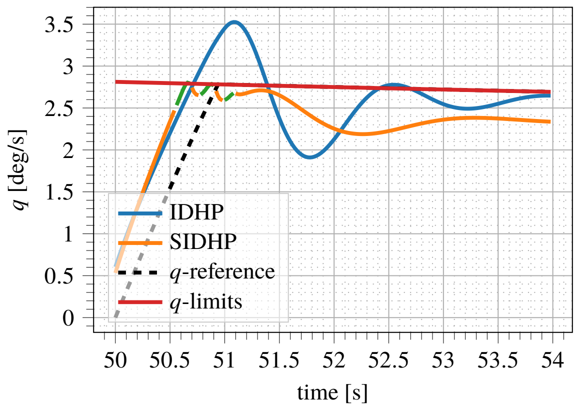

Fig. 7 Experiment 1: Enlarged view of Figure 6 showing a single run and zooming in on the boundary. Where SIDHP turns green the safety layer is activated.

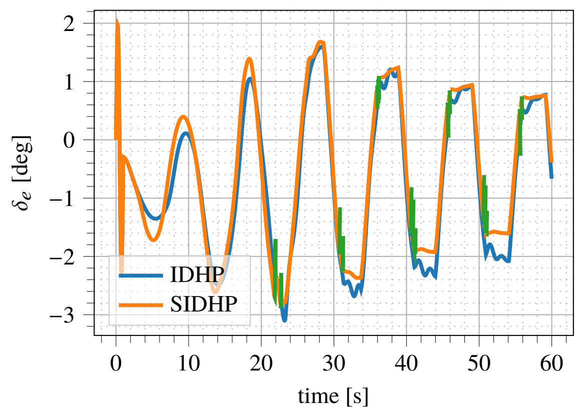

Fig. 8 Experiment 1: Elevator deflections for a single run. Where SIDHP turns green the safety layer is activated.

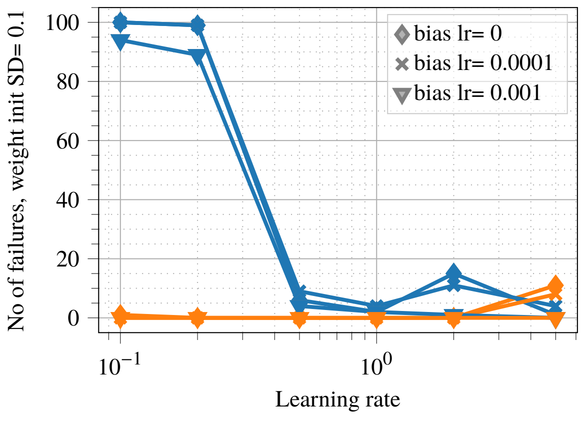

Fig. 9 Experiment 2: Number of failures for different combinations of learning rate and bias learning rate with a weight initialization of 0.1. Blue shows IDHP and Orange displays SIDHP. Lower is better.

Table 3 Overview of the limits used during the height tracking task

State	Lower Bound	Upper Bound
n [-]	-0.5	3
q [deg/s]	-15	20
α [deg]	-5	12
V [m/s]	80	130
H [m]	1800	2600

B. Experiment 2: Sensitivity Analysis

The robustness of reinforcement learning is often an issue. Convergence for a specific problem is often only achieved with specific learning rates and/or weight initializations whilst for a different problem these specific parameters could be far from ideal. For an aircraft, encountering a variety of different problems during its flight, this is undesirable. Therefore, in this experiment the robustness of IDHP and SIDHP with respect to their hyper parameters is compared. Both controllers have been applied to a height tracking task as seen in Figure 15. Each controller performs 100 runs for all combinations of hyper paramers as seen in Table 2. Additionally, limits have been set for n, q, α, V and H beforehand, which should not be crossed, representing a safe flight envelope that should be adhered to. These limits can be found in Table 3, it can be seen that the limits are now set beyond a normal flight envelope unlike in experiment 1. If these bounds are crossed the run is considered a failure.

In Figure 9, Figure 10 and Figure 11 the number of failed runs are shown with weight initializations of 0.1, 0.01 and 0.001 respectively. In these figures the x-axis shows the learning rate of the weights whilst the different markers show which specific bias-learning rate is used. In these figures 0 failures is optimal, whilst 100 failed runs is the worst performance achievable. The exact number of failures can be found in Table 4.

In all figures, and Table 4, it can be seen that with low learning rates IDHP does not converge in time and fails in (almost) every run. SIDHP however is also not able to converge quickly, but due to the safety layer and the accompanied learning impulse it manages to find a working policy and converge. An example of single run for both IDHP and SIDHP can be seen in Figure 15 and Figure 16 shows the elevator deflection given by IDHP and SIDHP. It can be seen that up until the convergence of SIDHP the safety layer makes small adjustment to ensure the given limits are not crossed. Although the controller is only able to control the inner loop pitch rate, the safety layer is able to ensure the outer loop altitude limits are not crossed. Using upset recovery strategies ensure that correction to the altitude only occurs if it is permitted according to the complete state. At low learning rates IDHP was unable to stay within the predetermined safe flight envelope in over 2300 out of 2700 runs, whereas SIDHP was unable stay within the safe flight envelope in only 3 out of 2700 runs.

For the data point with a learning rate of 1 IDHP and SIDHP perform similarly showing 0 failures for almost every combination of bias learning rate and weight initialization. In total IDHP exceeds the bounds in 39 out of 900 runs whilst SIDHP does so in 1 out of 900 runs. For a learning rate of 2 and 5 it can be seen that IDHP again is unable to find a working policy, except for the instance where the weight initialization was 0.1. SIDHP is also unable to attain low failure rate as the learning rate starts to increase. However the number of failures is still less than for IDHP. Again the exception is for the weight initialization of 0.1, here SIDHP actually creates more failures than IDHP. For the higher learning rates closer inspection showed that failures where mainly due to numerical errors. The high learning rate either ensured quick converge, or created an exploding gradient. The exploding gradient lead to infinite inputs of the elevator deflections which caused the simulation to fail. The safety layer is now effectively slowing down the learning process to ensure safe movements. Altough SIDHP was not created to combat these issues still a general improvement over IDHP can be seen.

In Figure 12, Figure 13 and Figure 14 the performance difference between IDHP and SIDHP is shown where the average of the square root of the cumulative reward of the successful runs is compared. The cumulative reward is defined according to Equation 8 and 9 and can be interpreted as the cumulative sum of the pitch rate error. This metric is comparable to the transient performance measure (TPM_2), which is proposed to be used in future aircraft certifications for adaptive control by [27]. The cumulative rewards are capped at -75, while 0 is the maximum achievable reward. When 100 failures occur the average reward is also set to -75.

It can be seen that IDHP and SIDHP are both unable to reach a reward above the threshold of -75 for low learning

Table 4 Overview of the number of failures for both IDHP and SIDHP categorized according to the learning rate.

Learning Rate	IDHP Failures	SIDHP Failures
0.1	894	1
0.2	887	2
0.5	597	0
1	39	1
2	361	98
5	387	227
Total	3165	329

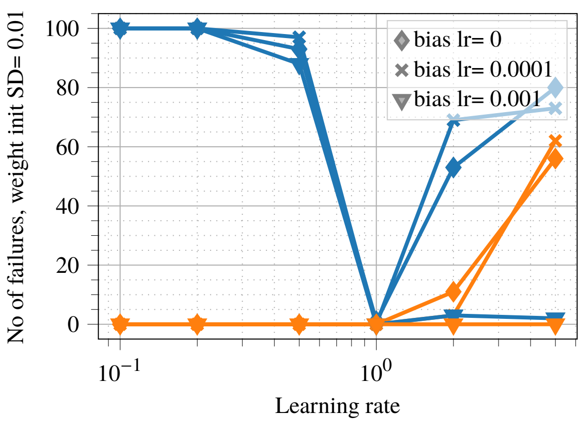

Fig. 10 Experiment 2: Number of failures for different combinations of learning rate and bias learning rate with a weight initialization of 0.01. Blue shows IDHP and Orange displays SIDHP. Lower is better

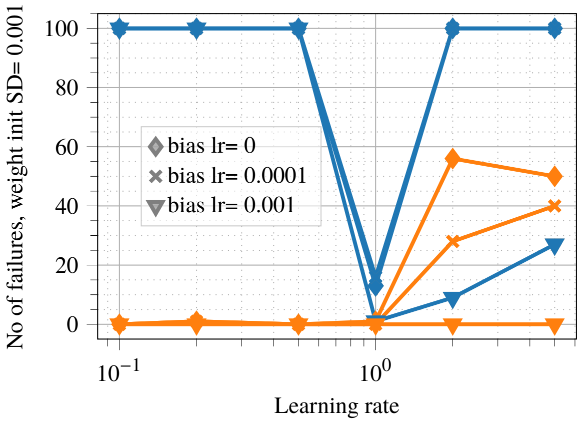

Fig. 11 Experiment 2: Number of failures for different combinations of learning rate and bias learning rate with a weight initialization of 0.001. Blue shows IDHP and Orange displays SIDHP. Lower is better

rates. As the learning rate improves and IDHP is able to complete a run its performance can be seen to be around 15% higher than for SIDHP. For higher learning rates the difference in performance is negligible. The fact that IDHP is not constrained in exploration can explain this increased performance. However, it is this freedom of exploration that also leads to failures.

In conclusion, it can be seen that SIDHP can be considered to be more robust to hyper parameters settings compared to IDHP. The number of failures is lower. The performance of IDHP is around 15% better on the successful runs, but only at low learning rates.

C. Experiment 3: Simulated Elevator Failure

This experiment displays the algorithms ability to deal with sudden changes in the environment. IDHP and SIDHP are both tasked with tracking a pitch rate reference similar to experiment 1. This time the limits on the pitch rate stay constant and the aircraft starts in trimmed condition. At 56 seconds a failure with the elevators is simulated. A loss of trim is experienced and the elevator effectiveness is decreased to 50%. In order to continue to fly the aircraft, the controllers have to both apply a different offset as the trim is lost, and increase the magnitude of command to compensate for the loss of elevator effectiveness. Again 100 runs are performed. In Figure 18 the average of 100 runs is shown, also displaying the minimum and maximum values encountered. Figure 19 zooms in on the behaviour at the time of the failure for a single run and in Figure 17 the elevator command is shown for a single run.

From the figures it can be seen that both controllers are able to converge online in a similar manner to a working control policy in around 30 seconds. When the failure occurs the aircraft drops in pitch rate due to the loss in trim. This

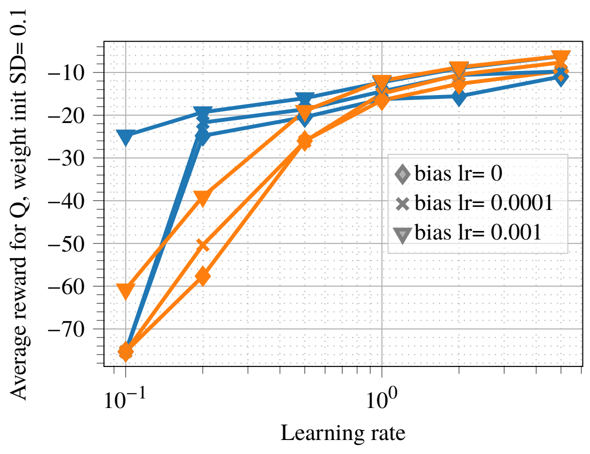

Fig. 12 Experiment 2: Average cumulative reward of the successful runs for different combinations of learning rate and bias learning rate with a weight initialization of 0.1. Blue shows IDHP and Orange displays SIDHP. Higher is better

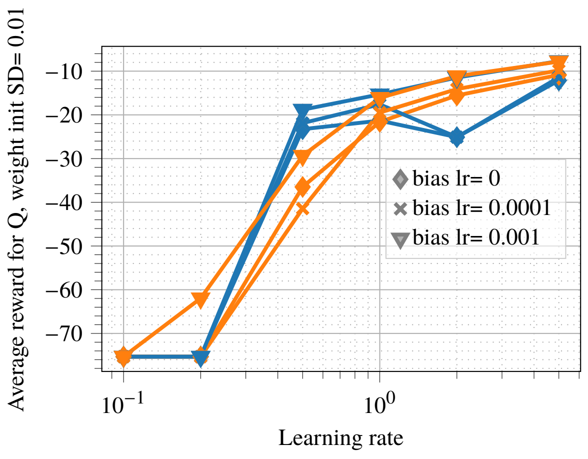

Fig. 13 Experiment 2: Average cumulative reward of the successful runs for different combinations of learning rate and bias learning rate with a weight initialization of 0.01. Blue shows IDHP and Orange displays SIDHP. Higher is better

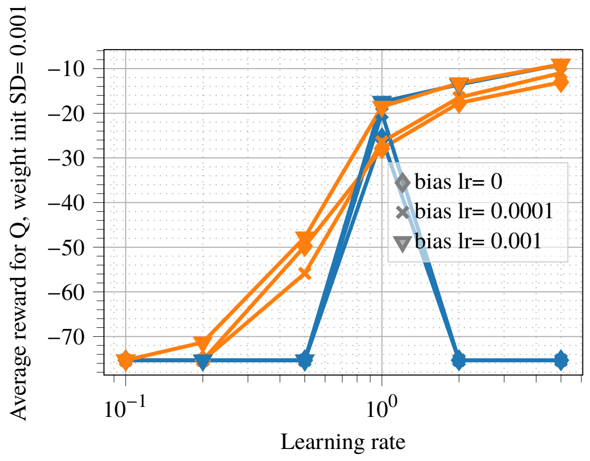

Fig. 14 Experiment 2: Average cumulative reward of the successful runs for different combinations of learning rate and bias learning rate with a weight initialization of 0.001. Blue shows IDHP and Orange displays SIDHP. Higher is better

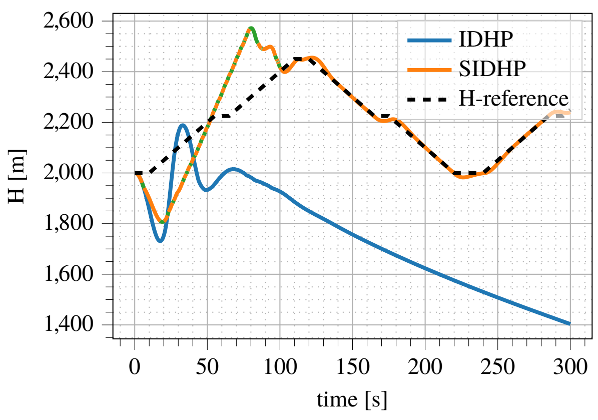

Fig. 15 Experiment 2: Example of a failed run using IDHP whilst the same settings resulted in a succesful run using SIDHP. Green indicates an activation of the safety layer.

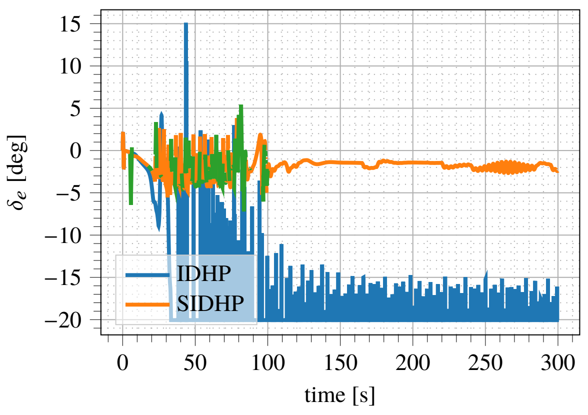

Fig. 16 Experiment 2: Elevator deflections corresponding to the example run shown in Figure 15.Green indicates an activation of the safety layer.

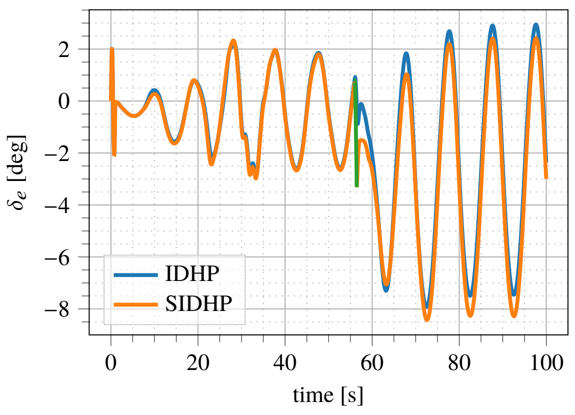

Fig. 17 Experiment 3: Elevator deflections outputted by IDHP and SIDHP. At 56 seconds the trim is lost and the elevator effectiveness is 50%. Green indicates an activation of the safety layer.

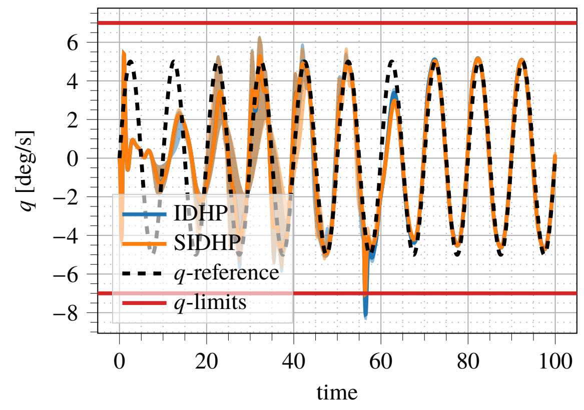

Fig. 18 Experiment 3: Tracking of a pitch reference rate q. At 56 seconds the trim is lost and the elevator effectiveness is reduced to 50%.

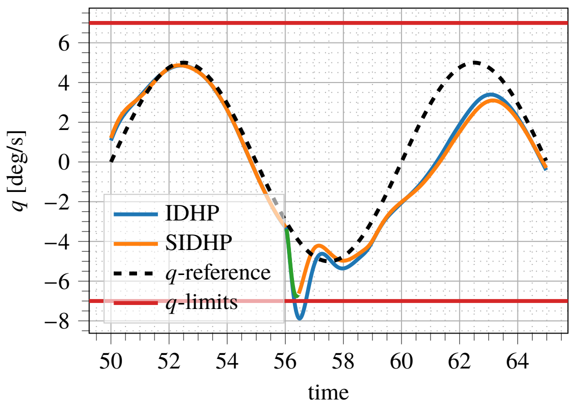

Fig. 19 Experiment 3: Enlarged view of Figure 18 showing the behaviour around the failure for a single run. Green indicates an activation of the safety layer.

is detected by both controllers and it can be seen, that after a short learning period and some corrective commands, the pitch rate reference is tracked again. Figure 17 shows how the offset has changed from around 0° to around -3° as a result of the loss of trim. Furthermore, it can be seen that the magnitude of the commands increase in order to deal with the loss of elevator effectiveness, as expected.

When zooming in on the behaviour near the failure occurrence, it can be seen that the safety layer almost instantaneously recognizes the failure. As a consequence a corrective action is proposed for SIDHP. This helps in evading the boundary and thus increases safety. The performance is very similar to IDHP demonstrating that the safety layer does not take away from the learning capabilities of the algorithm.

VI. Conclusion

The purpose of this research is to improve the robustness of continuous online reinforcement learning in flight control. SIDHP is presented as an improvement of IDHP, improving robustness and safety. This is done by adding a safety layer which uses predictions from a learned incremental model to alter actions which move the aircraft into a unsafe state. SIDHP is demonstrated on a high fidelity non-linear aircraft simulation of a Cessna Citation.

Experiment 1 proves the concept of SIDHP, showing how bounds set beforehand are not exceeded. Actions moving the aircraft outside the bound are corrected, increasing safety and performance. Experiment 2 compares the safety and robustness of SIDHP and IDHP on a height tracking task. It shows that SIDHP is more robust to changing hyper parameters. At low learning rates IDHP is unable to stay within the predetermined safe flight envelope in over 2300 out of 2700 runs, whereas SIDHP was unable stay within the safe flight envelope in only 3 out of 2700 runs. At the optimal learning rates for this problem the safety performance was similar. In total IDHP exceeds the limits in 3165 out of 5400 runs, compared to only 329 out of 5400 runs for SIDHP. Tracking performance on succesful runs is around 15% better for IDHP. Lastly experiment 3 demonstrates that the safety measures taken do not take away from the algorithms ability to adapt to sudden changes in the environment. After a simulated elevator failure both IDHP and SIDHP are able to converge to a new working policy. SIDHP is able to stay closer to the reference trajectory as a corrective action and learning impulse is provided by the safety layer.

Although more work is required, SIDHP shows a new method of introducing safety guarantees for the field of online continuous reinforcement learning in flight control. This results in a controller robust to aircraft damages or faults, but also to changing hyper parameter settings, allowing it to solve a wider range of problems. SIDHP provides a new solution that could allow the certification of reinforcement learning flight controllers on actual aircraft.

For further research it is recommended to investigate if the action corrective parameter can be adapted online using the incremental model. This is expected to improve safety at higher learning rates. Additionally, an adaptive learning rate can be implemented as has been done in [12]. Furthermore, this method could be combined with a risk signal, as used in [13], to not only correct the actions and provide the learning impulse when crossing of the bounds is imminent, but to get advance information about the bounds into the network. Lastly, in this research a safe flight envelope was identified beforehand, applying an online safe flight envelope identification system in combination with SIDHP would provide a complete solution.

References

[1] J. A. Stoop and J. P. Kahan. “Flying is the safest way to travel”. In: *European journal of transport and infrastructure research* 5.2 (2005), p. 115.

[2] D. Briere and P. Traverse. “AIRBUS A320/A330/A340 electrical flight controls - A family of fault-tolerant systems”. In: *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*. 1993, pp. 616–623.

[3] R. J. Adams and S. S. Banda. “Robust flight control design using dynamic inversion and structured singular value synthesis”. In: *IEEE Transactions on Control Systems Technology* 1.2 (1993), pp. 80–92.

[4] J. Doyle, K. Lenz, and A. Packard. “Design examples using μ-synthesis: Space shuttle lateral axis FCS during reentry”. In: *Modelling, Robustness and Sensitivity Reduction in Control Systems*. Springer, 1987, pp. 127–154.

[5] B. Kulcsár. “LQG/LTR Controller design for an aircraft model”. In: *Periodica Polytechnica Transportation Engineering* 28.1-2 (2000), pp. 131–142.

[6] J. Reiner, G. J. Balas, and W. L. Garrard. “Robust dynamic inversion for control of highly maneuverable aircraft”. In: *Journal of Guidance, control, and dynamics* 18.1 (1995), pp. 18–24.

[7] S. Sieberling, Q. P. Chu, and J. A. Mulder. “Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction”. In: *Journal of Guidance, Control, and Dynamics* 33.6 (Nov. 2010), pp. 1732–1742.

[8] G. J. Balas. “Flight Control Law Design: An Industry Perspective”. In: *European Journal of Control* 9.2-3 (2003), pp. 207–226.

[9] F. Grondman et al. “Design and Flight Testing of Incremental Nonlinear Dynamic Inversion-based Control Laws for a Passenger Aircraft”. In: *2018 AIAA Guidance, Navigation, and Control Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics, Jan. 2018.

[10] T. P. Lillicrap et al. *Continuous control with deep reinforcement learning*. Sept. 2015. URL: `http://arxiv.org/abs/1509.02971`.

[11] T. Bansal et al. *Emergent Complexity via Multi-Agent Competition*. Oct. 2017. URL: `http://arxiv.org/abs/1710.03748`.

[12] Y. Zhou, E. J. van Kampen, and Q. P. Chu. “Incremental model based online dual heuristic programming for nonlinear adaptive control”. In: *Control Engineering Practice* 73.July 2017 (2018), pp. 13–25.

[13] T. Mannucci et al. “Safe Exploration Algorithms for Reinforcement Learning Controllers”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.4 (2018), pp. 1069–1081.

[14] R. M. Kretchmar et al. “Robust Reinforcement Learning Control with Static and Dynamic Stability”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 11.15 (2001), pp. 1–25.

[15] J. Garcia and F. Fernandez. “A Comprehensive Survey on Safe Reinforcement Learning”. In: *The Journal of Machine Learning Research* 16 (2015), pp. 1437–1480.

[16] G. Dalal et al. “Safe Exploration in Continuous Action Spaces”. In: *arXiv preprint arXiv:1801.08757* (2018).

[17] S. Jacklin. “Closing the certification gaps in adaptive flight control software”. In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2012, p. 6988.

[18] G. Brockman et al. *OpenAI Gym*. June 2016. URL: `http://arxiv.org/abs/1606.01540`.

[19] S. Heyer. *Reinforcement Learning for Flight Control: Learning to fly the PH-LAB*. Delft, University of Technology, 2019.

[20] D. Kroezen, E.-J. van Kampen, and Q. Chu. *Online Reinforcement Learning for Flight Control*. Delft, University of Technology, 2019.

[21] S. Gu et al. *Continuous Deep Q-Learning with Model-based Acceleration*. Mar. 2016. URL: `http://arxiv.org/abs/1603.00748`.

[22] R. S. Sutton and A. G. Barto. *Reinforcement Learning An Introduction*. 2nd. Cambridge, MA, USA: MIT Press, 2018.

[23] Schaal, S. “Learning from demonstration”. In: *Advances in Neural Information Processing Systems* 9 (1997), pp. 1040–1046.

[24] F. Berkenkamp, A. P. Schoellig, and A. Krause. “Safe controller optimization for quadrotors with Gaussian processes”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. June. 2016, pp. 491–496.

[25] R. Collinson. “Fly-by-wire flight control”. In: *Computing & Control Engineering Journal* 10.4 (Aug. 1999), pp. 141–152.

[26] S. Bhattacharyya et al. *Certification Considerations for Adaptive Systems*. Tech. rep. 2015.

[27] V. Stepanyan et al. “Stability and Performance Metrics for Adaptive Flight Control”. In: *AIAA Guidance, Navigation, and Control Conference*. Reston, Virigina: American Institute of Aeronautics and Astronautics, Aug. 2009, pp. 1–19.

[28] L. Crespo et al. “Analysis of control strategies for aircraft flight upset recovery”. In: *AIAA Guidance, Navigation, and Control Conference*. 2012, p. 5026.

[29] C. van der Linden. *DASMAT-Delft University aircraft simulation model and analysis tool: A Matlab/Simulink environment for flight dynamics and control analysis*. Delft, University of Technology, 1996.

[30] 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*. 2018, pp. 259–277.

[31] Y. Zhou, E. van Kampen, and Q. Chu. “Autonomous navigation in partially observable environments using hierarchical q-learning”. In: *Proceedings of the 2016 International Micro Air Vehicles Conference and Competition, Beijing, China*. 2016.

II

Literature and Preliminary Analysis

All chapters in Part II have been graded during the Literature Study.

2 Literature Overview

In order to familiarize ourselves with the topic of safe online continuous reinforcement learning an overview of the performed literature study is given in this chapter. First the topic of reinforcement learning is introduced in section 2.1, followed by an overview of the current state-of-the-art in (safe) reinforcement learning shown in section 2.2. Lastly the topic of safety in flight control is discussed in section 2.3.

2.1. Introduction to reinforcement learning

The purpose of this section is to introduce the topic of reinforcement learning. This is done by first providing a short history of reinforcement learning. Next an overview of the methodology used is given. It should be noted that most information in this chapter is taken from ”Reinforcement Learning: an Introduction” [65].

2.1.1. Short history of Reinforcement Learning

According to Sutton and Barto [65] the history of reinforcement learning can be separated into three separate threads that later combined, formed reinforcement learning as we now know it. One thread was concerned mostly with learning via trial and error and is visible in the earliest days of artificial intelligence. The second thread focuses on the problem of optimal control using value functions and dynamic programming and did not involve learning. The last thread using temporal difference methods (see subsection 2.1.2) developed itself parallel to the first thread but only found its way into reinforcement learning at the end of the 1980’s. In the 1980’s all threads combined to form what is now regarded as the field of reinforcement learning.

Trial and error learning, finds its roots in animal learning. Thorndike worded the ”Law of Effect” to describe the effect of reinforcing events after actions were taken [67]. Pavlov’s research on conditioned reflexes first contained the term ”reinforcement” to describe how a pattern or behaviour is strengthened based on a reward or punishment. Trial and error learning in a computer was also already mentioned in the earliest ideas for artificial intelligence. Alan Turing describes a ”pleasure-pain system” which learns based on a feedback signal [68] where a pain signal cancels all behaviour and a pleasure signal makes the behaviour permanent. In 1950 W. Walter demonstrated his ”mechanical tortoise” which are tortoise shaped robots capable of simple learning [69] through analog components and sensors. The idea of using a brain-like computer structure can already be found in Marvin Minsky’s Phd Thesis, here he used analog components to build a network of synapses that resembles the synapses in the brain [51]. The machine could alter its connection based on a reward signal. He also used the term reinforcement learning in engineering literature to describe trial-and-error learning.

The coming forth of digital computers allowed programming of various learning tasks. However pure trail-and-error learning, that is learning on the basis of evaluative feedback, became rare during the 1960’s and 70’s. This can be attributed to the rise of supervised learning and a confusion that arose about the relationship of supervised learning and reinforcement learning. Many researchers used terms

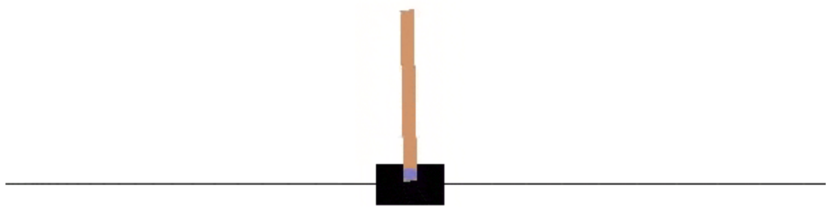

Figure 2.1: Modern implementation of the cartpole task as used by Michie and Chambers. The goal is to balance the pole by controlling the cart by means of force inputs left or right. The task terminates when the pole falls or the cart moves off the rail.

such as reward and punishment, reinforcement learning terms, whilst their systems were supervised systems or included an all-knowing teacher within their systems.

In 1961 Michie presented a reinforcement learning controller MENACE which was able to learn how to play tic-tac-toe through reinforcement learning. Later in 1968 Michie and Chambers presented GLEE which could also master tic-tac-toe and BOXES. BOXES was applied on a pole balancing task hinged on a moving cart with a failure signal giving feedback when the pole fell or the cart reached the end of the track [50]. A modern implementations of the cartpole task can be seen in Figure 2.1. This can be seen as one of the best early examples of reinforcement learning, the controller does not know anything beforehand and learns through experience. It is also an early example of reinforcement learning being used in control task and not just to play a game.

In 1972 Harry Klopf realized that the adaptive behaviour of reinforcement learning was lost as research focused more and more on supervised learning. As a results he focused on trial-and-error learning and opted for a more hedonic approach [40]. Later Sutton and Barto provided a clear separation between supervised learning and reinforcement learning [8].

The second thread describes the problem of designing a optimal controller. In 1950 Richard Bellman and others extended the Hamilton Jacobi equation and created the Bellman equation. Here an optimal return function is used to define what we now know as the Bellman equation. When solving problems using the Bellman equation it is known as dynamic programming. He also introduced a discrete version which we know now as a Markov Decision Process (MDP) [10]. Much research has been done on the subject of dynamic programming, but it was not until 1987 that Werbos proposed to combine dynamic programming with learning methods [72]. Next Watkins used reinforcement learning in combination with MDP's in order to create an online learning method using dynamic programming [71]. From this point the connection between reinforcement learning and dynamic programming has only grown.

The third thread concerning temporal difference learning also find its roots in animal learning psychology. Minsky realized early on that this principle could also be important in artificial intelligence. Next Arthur Samuel implemented a learning method that contained several temporal difference ideas in 1959. Then for a long time, as with trial-and-error learning, no work was done on the subject of temporal difference learning until Harry Klopf revived the idea in 1972. Sutton and Barto, who were inspired by Klopf and animal learning theories, then developed a model that included temporal difference learning. They developed the actor-critic architecture and applied to the same pole-balancing task Michie and Chambers used [9].

When Watkins started developing Q-learning in 1989 all threads described above were joined together [70]. Continued development has created reinforcement learning agents that are capable of outperforming humans in simple computergames and more complex tasks such as Chess or Go [61].

2.1.2. Methodology of Reinforcement Learning

To introduce the methodology of reinforcement learning this section will first present the basic idea driving reinforcement learning. This followed by discussing different methods found within the field of reinforcement learning.

”Reinforcement learning is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal [65].” Actions may have effect not only on the immediate reward, but also on future rewards. An algorithm, or agent, must learn from its own experience and is not told what action to take or what reward can be expected. In this research the FCS takes the role of agent and must choose which action to take such that the aircraft behaves as desired. The taken action will have an effect on the environment, consisting of the full aircraft dynamics, and in turn the agent receive the states and reward signal as feedback to close the loop. From these states the agent will choose a new action again. This is displayed in Figure 2.2. From experience the agent must learn what actions is best given a certain state. This is done by maximizing the cumulative rewards received, where the rewards signal can be seen as a rating for the state the agent is in. Using a reward signal to reach a capable controller is a key concept of reinforcement learning. The reward signal should therefore be designed such that it guides the agent towards the desired controller. For example if we want an agent to fly an aircraft and follow a certain trajectory we can design the reward signal such that it receives a reward of 1 when it follows the trajectory and a reward of 1 minus the tracking error when the aircraft deviates. In the long run the agent will receive the maximum reward when it follows the trajectory precisely and will learn to do so.

For simple problems with limited states information regarding what action is best given a certain state can be stored in a simple look-up table, but for more complex problems these tables would require an immense amount of memory due to the ”curse of dimensionality” and thus approximations are used. In this research these approximations are created using neural networks. This also has the extra benefit that the neural network can approximate from experience what the optimal action is in a state it has never encountered. These look-up tables and networks can then be trained using different reinforcement learning algorithms and functions.

In this section the concepts of value functions and policy functions are explained. This is followed by a short overview of the basic learning methods most often used, namely Dynamic Programming, Monte Carlo learning and Temporal Difference learning. These learning methods will be discussed based on a discrete finite problem which can be solved using a look-up table. Next continuous learning is discussed and the use of approximations is presented, this leads to a solution also capable of solving continuous problems.

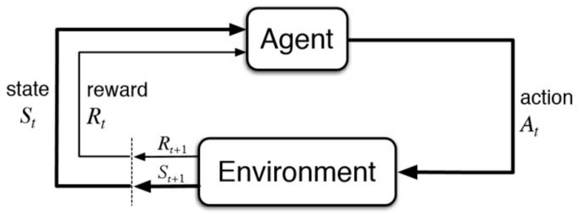

Figure 2.2: Overview of a simple Reinforcement learning setup.

Policy function

A policy is a stochastic rule by which the agent selects actions as a function of the current states [65]. It decides what action to take not based on the expected return, but based on its current state. For example when driving a car and the light turns orange a policy could be: ”I will stop when I am more than 30 meters from the light.” In Equation 2.1 it can be seen that a policy defines the conditional probability of each action given a certain state.

$$\pi(a|s) = Pr\{A_t = a|S_t = s\} \tag{2.1}$$

A special case of a policy is a greedy policy. It only chooses the action that has the highest probability of reward and can be considered an optimal policy. When an agent applies such method it will never visit states with low probability, however it can thus also not experience subsequent states that possibly lead to higher reward. Thus from a learning perspective a greedy policy is not optimal and exploration of all states is required.

Value function

The goal of the learning agent is to maximize the cumulative rewards it receives in the long run. In literature this is referred to as return, G_t, given in Equation 2.2. Here R_{t+1}, R_{t+2}, R_{t+3},.. indicate the rewards received after timestep t and R_T is the reward in the final timestep.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T \tag{2.2}$$

Equation 2.2 works well when a final timestep exist and training can be performed episodically. However many tasks are continuous and can not be broken up into episodes. This will lead to infinite rewards disrupting the learning process. Therefore future rewards are discounted using a discount factor, γ, which varies between 0 and 1. If γ is closer to 0 the agent will focus on immediate returns, when γ is closer to 1 the agent will look more into the future when deciding which action to take. Equation 2.3 and Equation 2.4 show how the discounted return function is defined where Equation 2.4 can be easier to implement in algorithms.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{2.4}$$

Value functions estimate how good a specific state is or how good a specific action in a given state is. This is done by assigning to each state the expected total return, where future rewards are usually discounted, based on the current policy. Equation 2.5 shows the definition of an on-policy value function, meaning that starting from state s, policy π is followed resulting in the state-value function.

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | St = s\right] \tag{2.5}$$

It is also possible to use a so called off-policy function or action-value function as shown in Equation 2.6. Here the expected return is calculated given a certain state s and taking action, a, possibly differing from policy π, and thereafter following policy π.

$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | St = s, A_t = a\right] \tag{2.6}$$

The state-value and action-value function can be estimated from experience. During training, experience refines the value function and it increases in accuracy. When using look-up tables the values in the table are some sort of mean value (depends on the method) of all previous decisions. When an approximation function is used the parameters of the function are altered with each update as to ”fit” all measurements.

Dynamic Programming

Although Dynamic programming (DP) is at the basis of reinforcement learning as it used today, DP in itself is less applicable to most reinforcement learning problems. This is due to the fact that DP assumes that a perfect model of the environment is known which, in the case of reinforcement learning, is usually not the case. Furthermore DP can have a significant computational expense.

DP can be seen as iterative loop, where first the value function is predicted, then the policy is altered such that it is greedy with respect to the new value function. Next the value function is again estimated with the new policy and so forth until the solutions converge. Starting from policy π_0 this can be visualed, where π_* and v_* indicate the optimal policy and optimal value function respectively:

$$\pi_0 \rightarrow v_{\pi_0} \rightarrow \pi_1 \rightarrow v_{\pi_1} \rightarrow \dots \rightarrow \pi_* \rightarrow v_*$$

.

In DP finding the value function is known as policy evaluation. This is done according to Equation 2.7 where we start from Equation 2.5. Here $p(s', r|s, a)$ indicates the probability of ending up in state s'

and receiving reward r when action a is taken in state s. DP assumes this knowledge is known for all states and actions as was stated above.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\big[G_t | S_t = s\big] = \mathbb{E}_\pi\big[R_{t+1} + \gamma G_{t+1} | S_t = s\big] \\ &= \mathbb{E}_\pi\big[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\big] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\big[r + \gamma v_\pi(s')\big] \end{aligned} \tag{2.7}$$

In practice this is done using iterative policy evaluation. An array $V(s) = 0$ is initialized for all states and the current policy is used as input. Next for each element in $V(s)$ the value is calculated using Equation 2.7. When the updates are sufficiently small or $|v_{k+1}(s) - v_k(s)| < \theta$, where θ is a threshold set in advance, the loop is halted.

With the value function known the policy can be improved known as policy improvement. For this use is made of the action-value function as defined in Equation 2.6, this leads to Equation 2.8

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi\big[G_t | S_t = s, A_t = a\big] = \mathbb{E}_\pi\big[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a\big] \\ &= \sum_{s'} \sum_r p(s', r|s, a)\big[r + \gamma v_\pi(s')\big] \end{aligned} \tag{2.8}$$

As the action-value function determines the value for an action, $a \neq \pi(s)$, it can then be compared to the value function. If $q(s,a) > v_\pi(s)$ the selected action yields more reward and it is thus better to select action a instead of following $\pi(s)$. When this is extended to all possible states and actions Equation 2.9 defines the new policy π'. Argmax$_a$ indicates the value a where the expression is maximized, such that the policy is greedy with respect to the value function.

$$\pi'(s) = \operatorname{argmax}_a q_\pi(s,a) \tag{2.9}$$

$$= \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r|s, a)\big[r + \gamma v_\pi(s')\big] \tag{2.10}$$

With the improved policy now a new value function needs to be found. Policy iteration combines iterative policy evaluation and policy improvement looping over them until the optimal policy and optimal value function is found. This is displayed in Figure 2.3. When the new greedy policy gives the same value function as the old policy, or $v\pi = v'_\pi$, v_π and v'_π are equal to the optimal value function. This also means that π and π' are both optimal policies.

Different variants of DP exist each, but all make use of Generalized Policy Iteration (GPI). This term defines the general idea of letting the policy evaluation and policy improvement interact to obtain the optimal value and policy function. As both functions are improved the process stabilizes and thus the optimal functions are reached.

Monte-Carlo learning

Unlike DP Monte-Carlo (MC) learning methods do not need any knowledge or model to reach a solution. MC is based purely on experience, either actual experience or simulated experience, and is still capable of reaching the optimal policy. It solves reinforcement learning problems based on averaging the sampled returns. MC methods only works for episodic tasks where each episode eventually terminates. Just as with DP, MC methods make use of GPI. After an episode terminates the experience is used to update the value estimation. This is then used to obtain a new policy.

In order to estimate the value function from experience all returns obtained after visiting a certain state for the first time are averaged, as more returns are observed, the average will converge towards the expected value following the Law of large numbers (alternative strategies exist, but for simplicity only the first-visit MC method is used to illustrate the principle). However as a model is not available in most

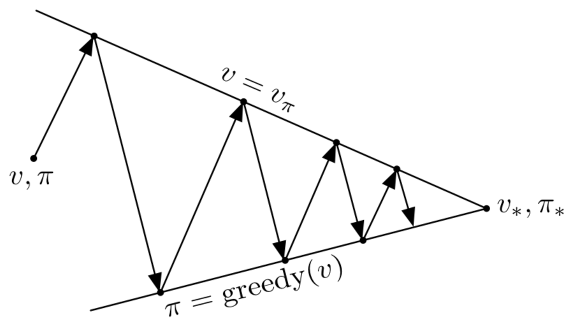

Figure 2.3: Policy iteration is shown as an interaction of policy evaluation and policy improvement starting with a random value and policy. [65]

problems state values are not sufficient. One must look at the action values in order to be useful in suggesting a policy as it impossible to simply look one step ahead and choose the action corresponding to the highest value without a model. Therefore MC methods try to estimate $q_\pi(s,a)$ where s,a is a state action pair. Each state action pair will have to be visited an infinite amount of times to converge to the expected value. This leads to a problem as many state action pair are not visited if a deterministic policy is used. In order to maintain exploration each state action pair should have a non-zero probability of being selected. Alternatively the policy can be stochastic resulting in all state action pairs being visited. Equation 2.11 shows how a single action value is obtained, where G_t are the returns after the first visit of the state-action pair. As the number of visits N increase the estimated value approaches the true action value. Equation 2.12 shows an incremental version that is easier to implement. Here α defines the step-size and can be constant, but also vary as the number of updates increase.

$$Q(s,a) = \text{mean}\Big((G_0|S_t = s, A_t = a), (G_1|S_t = s, A_t = a), \ldots, (G_N|S_t = s, A_t = a)\Big) \tag{2.11}$$

$$Q(s,a) = Q(s,a) + \alpha\big[G_t - V(S_t)\big] \tag{2.12}$$

Through the estimated action value function an improved policy can be found. This is done by making the policy greedy with respect to the action value function. A greedy policy, shown in Equation 2.13, is a policy that always chooses the action, a, that has the highest action value in the given state, s. In Equation 2.14 the new greedy policy π_{k+1} is shown to always be better or equal to the previous policy. If the policies are performing just as well this shows both policies are optimal policies just as with DP.

$$\pi(s) = \text{argmax}_a q(s,a) \tag{2.13}$$

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \text{argmax}_a q(s,a)) \\ &= \text{max}_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \end{aligned} \tag{2.14}$$

Temporal Difference learning

Temporal Difference (TD) learning combines the advantages of both DP and MC. It uses bootstrapping just as DP does (TD updates its estimates based on other learned estimates) and TD is capable of learning directly from experience and thus does not require a complete model as is the case with MC. Additionally TD is an online method, meaning that adjustments can be made during an episode. In practice this usually results in faster convergence when compared to MC in a similar setting. A mathematical prove of TD outperforming MC has yet to be given.

The benefit of bootstrapping can be illustrated using a car journey as an example. Lets say a standard journey takes around 30 minutes. If we get stuck in a traffic jam with an expected delay of 15 minutes a TD method can immediately change its journey time to 45 minutes. From this estimate we can then determine whether an diversion would now be beneficial (i.e. changing the policy). When MC is used we first have to arrive at our destination before the update can be made and no diversions are made.

As bootstrapping uses estimates to learn a new estimate a guarantee of convergence is needed. In [64] this was given for the case of tabular learning.

As no model is used the action value needs to be estimated as was the case with MC. This is done according to Equation 2.15 when the SARSA method is used and according to Equation 2.16 when Q-learning is applied. As can be seen Q-learning and SARSA both use the estimate value of the next state-action, the only difference being that Q-learning always takes the maximum of the next possible state-action pair estimate values regardless which state-action pair is actually visited whilst SARSA uses the estimates value of the state-action pair that is visited. This makes SARSA an on-policy learning method and Q-learning an off-policy learning method Equation 2.11 and Equation 2.12 showed the action value when using MC and are interesting to compare.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{2.15}$$

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{2.16}$$

TD also used GPI in order to obtain both the optimal value function and the optimal policy function in a similar fashion as MC does. Therefore this explanation will not be given again here. Throughout reinforcement learning TD methods are often used due to their simple model-free online implementation. Various algorithms with various extensions exist that make TD even more powerful. All algorithms however make use of what is known as the TD error defined in Equation 2.17.

$$\delta_t = r_t + \gamma V_{s_{t+1}} - V_{s_t} \tag{2.17}$$

Continuous Learning Methods

One of the big challenges in reinforcement learning is the amount of simulation that needs to be performed in order for the agent to converge to a solution. All state-action pairs have to be visited many times before the solution converges. As the state space and action space increases the amount of simulation required also increases. In order to use continuous state- and action-spaces and allow for faster convergence faster convergence an approximation of the policy function can be used, or so called policy gradient method. Different types of approximations can be used, but in this research each function is approximated using a neural network that maps the respective input to the respective output. The policy function is now a parameterized policy, where the parameters θ are the weights of the neural network, which can select an action without consulting the value function. This can be written as $\pi(a|s,\theta) = Pr(A_t = a|S_t = s, \theta_t = \theta)$. This greatly increases the convergence speed as the parameterization also has a learned solution for state-action pairs not visited. Therefore continuous state spaces and action spaces can also be solved.

As now a parameterization is used instead of a lookup-table the update process is slightly different. First a performance measure is needed with respect to the policy parameter, $J(\theta)$. The policy can then be optimized using Equation 2.18. Here the expectation is used as the experience gathered only approximates the true gradient. Which performance measure is used differs per algorithm.

$$\theta_{t+1} = \theta_t + \alpha\mathbb{E}\left[\nabla J(\theta_t)\right] \tag{2.18}$$

Examples of policy gradient methods are REINFORCE and Actor-Critic learning. REINFORCE uses MC learning and can thus typicaly only be applied on episodic task and offline learning. Actor-Critic learning on the other hand makes use of TD learning which reduces variance and increases convergence speed. In order to estimate the value function also an approximation is used which is used to assist in training the policy function. Actor-Critic learning has received more attention and many further developments when compared to REINFORCE.

The method used to by actor-critic algorithms is best shown in Figure 2.4. It can be seen that the environment outputs a state and a reward. The value function, or critic, uses this state to give a value estimation. It can then subsequently use the reward signal to train itself on providing the best value estimation possible. The value estimation outputted by the critic can then be used to train the policy function, or actor, by using the TD error given in Equation 2.17. The actor is given the state of the environment and is tasked with mapping it to an action. This action is then send to the environment to complete the loop.

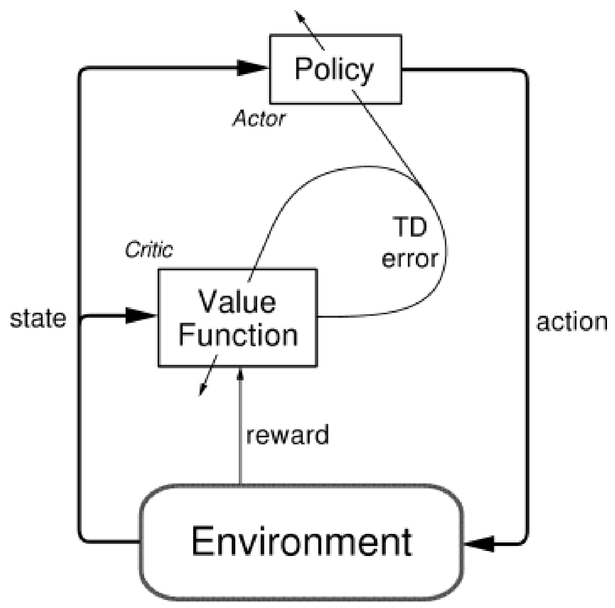

Figure 2.4: Overview of a generic Actor-Critic algorithm. [65]

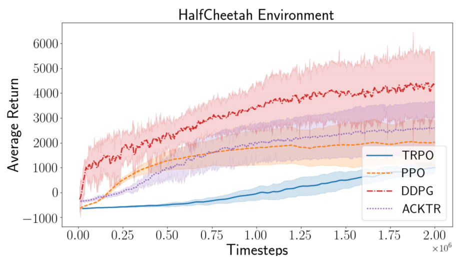

Figure 2.5: Rewards obtained during training process of different algorithms. DDPG is shown to outperform other methods on this particular environment.[31]

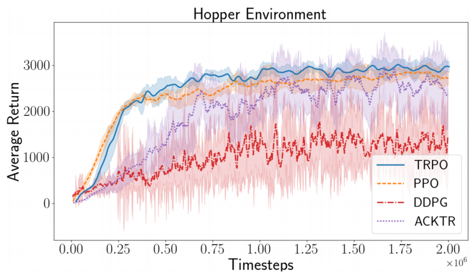

Figure 2.6: Rewards obtained during training process of different algorithms. PPO is shown to perform similar to other state of the art methods besides being much simpler.[31]

2.2. State-of-the-Art and Safe Reinforcement Learning

In this section first some state-of-the-art RL algorithms used for control tasks are presented that could benefit this research. Next the state-of-the art in safe reinforcement learning is discussed.

2.2.1. State-of-the-Art Reinforcement Learning

Since Michie first used reinforcement learning to control a process [50] much development and research has been performed on the topic of reinforcement learning. Reinforcement learning has been applied in a wide range of control tasks such as self driving cars, robot locomotion, quadcopter control and helicopter control [29, 14, 53, 49]. Furthermore reinforcement learning has also proven itself in other fields such as strategy games, resource managements, etcetera. In general research has lead to highly optimized algorithms using Artificial Neural Networks (ANN) as approximation function to solve complex situations. First Deep Deterministic Policy Gradient (DDPG) is outlined followed by Normalized Advantage Function (NAF), the only non actor-critic method in this section. Furthermore the Proximal Policy Optimization (PPO), Dual Heuristic Programming (DHP) and Incremental model based Dual Heuristic Programming (IDHP) algorithms are presented. The choice for these algorithms is made as these have either progressed the field of reinforcement learning or seem applicable to adaptive flight control.

Deep Deterministic Policy Gradient (DDPG)

DDPG [44] is the first actor-critic model-free algorithm using deep neural networks as approximation functions capable of working with continuous action spaces. DDPG combines the actor-critic approach with insights from DQN [52]. As such it makes use of TD learning and policy gradient methods as described in subsection 2.1.2. This resulted in an algorithm that is shown to robustly solve over 20 simulated physics tasks ranging from the classic cartpole tasks to legged locomotion. Furthermore the amount of samples required for training proved to be a factor 20 less than with DQN, but still required up to 2.5 million steps of experience. In Figure 2.5 DDPG is shown to outperform much more recent algorithms, Figure 2.6 on the other hand shows that on a different task DDPG is outperformed by the same algorithms.

The authors firstly attribute the success to training off-policy with a replay buffer. This entails storing all experience in a buffer and learn from this experience by random sampling out of the buffer. Using a replay buffer is used to minimize correlations between samples. Secondly updating the target Q network is done with so called ”soft” target updates in order to stabilize the learning process. Next they name batch normalization, were all states are normalized to account for the fact that states often have different units. Lastly the authors encouraged exploration by adding a noise parameter to the actor policy which was possible as off-policy algorithm was used. The critic network is updated using Equation 2.19 as loss function, Equation 2.20 shows how the gradients for the actor are estimated. Where μ, θ^Q and θ^μ represent the actor network function, the weight of the critic network and the weight of the actor network respectively. δ_i is the TD error as shown in Equation 2.17.

$$L = \frac{1}{N}\sum_i (\delta_i - Q(s_i, a_i|\theta^Q))^2 \tag{2.19}$$

$$\nabla\theta^{\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a, |\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla\theta^{\mu}\mu(s|\theta^{\mu})|s_i \tag{2.20}$$

(2.21)

Normalized Advantage Function (NAF)

NAF, as presented in [27], can be seen as a continuous version of the Q-learning algorithm by using function approximators. Furthermore the authors have studied the impact of implementing known model knowledge into the algorithm. Since it is not an actor-critic method NAF is considerably easier to implement in comparison to the other algorithms presented here.

In NAF a neural network is used to output the value function term $V(s)$ and an advantage term $A(s, a)$. These are consequently used to determine the action and also needed to provide network updates. Optionally a linear model is iteratively refitted and allows for off-policy training, simultaneous to on-policy application on the simulator. The authors have also incorporated experience replay and soft target updates as was the case with DDPG.

NAF is shown to outperform DDPG for different tasks. The use of learned models allows for faster training, but it only proves to be significant during the first phase of the learning process. The authors further state that it was imperative for the algorithm to also experience unsuccesfull actions as this builds an accurate estimate of the Q-function.

Proximal Policy Optimization (PPO)

In 2015 TRPO [59] was presented which proved to outperform other state-of-the-art RL algorithms. This was done by analytically proving a monotonic improvement using a Kullback–Leibler divergence which stabilizes the training. Unfortunately TRPO also proved to be computationally expensive. PPO [58] tries to maintain the advantages of TRPO whilst being less computationally expensive and easier to implement. This is done by increasing the training stability through clipping the probability ratios within the objective functions. This allows for training with multiple epochs on training data, something that has not been accomplished before. Furthermore an advantage function is used as described in [57], which can be seen as a measurement of how much better or worse the chosen action was than expected. Combined this leads to the loss clipped loss function described in Equation 2.22, where $r_t(\theta)$ is given in Equation 2.23.

$$L_{\text{clip}} = \hat{E}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right] \tag{2.22}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t} \tag{2.23}$$

PPO is also capable of running multiple simulations in parallel, according to the authors this actually increases performance as the correlation within the experience is lessened. Indeed PPO proved to perform just as well as TRPO, but with less sample complexity and decreased wall clock time. Furthermore implementing PPO is much simpler. Again Figure 2.5 and Figure 2.6 show benchmarks that compare PPO, TRPO and DDPG. ACKTR also shown in the figures can be ignored for this study.

Dual Heuristic Programming (DHP)

The state-of-the-art algorithms described above often need many trials before a working policy is found. They can be applied to complex problems, starting with no knowledge of the environment at all and use Deep Reinforcement Learning (DRL) to find solutions. Another area of research called Approximate Dynamic Programming (ADP) uses Adaptive Critic Designs (ACD) to find solutions specifically for non-stationary control problems. In order for this method to work a model of the environment is required to enable feedback to the critic and/or actor. On the plus side these methods converge quickly and can be adapted online (after an initial offline training). Using a model to speed up learning has recently also been applied on deep reinforcement learning with promising results [26, 46].

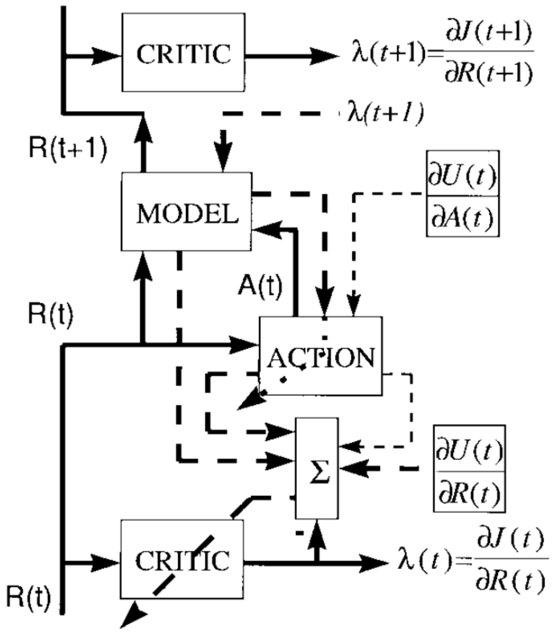
CRITIC
$\lambda(t+1)=\frac{\partial J(t+1)}{\partial R(t+1)}$
$\lambda(t+1)$
R(t+1)
MODEL
$\frac{\partial U(t)}{\partial A(t)}$
A(t)
R(t)
ACTION
$\frac{\partial U(t)}{\partial R(t)}$
Σ
CRITIC
$\lambda(t)=\frac{\partial J(t)}{\partial R(t)}$
R(t)

Figure 2.7: Overview of the DHP algorithm flow [54]. The 2 critic's displayed are actually the same critic, but at different timesteps.

Multiple variants of ACD exist, with Heuristic Dynamic Programming (HDP) being one of the most known variants. HDP is later extended to DHP which up to now proves to be the best performing variant. In DHP the critic is used to estimate the partial derivatives of the value function to each state instead of the value function itself as is the case with HDP [54]. These derivatives are required for training the actor. Approximating the derivatives instead of approximating the value function and then obtain the approximated derivatives through back-propagation leads to a better quality in estimation. In Figure 2.7 the flow of the DHP method is shown, it can be seen that a model of the environment is required in order to back-propagate the gradients to the actor and critic. The derivative of temporal difference error is used as a loss function for the critic, whilst the actor tries to maximize the value function.

Incremental Heuristic Dynamic Programming (IHDP)

Although its adaptive behaviour and short training are very desirable, the dependency on an accurate model is a drawback of using DHP. IHDP replaces the model with a estimated incremental model which is found using recursive least squares[76]. This allows for near optimal control without any prior knowledge of the system dynamic. Furthermore the offline training phase has also been made obsolete.

The authors have shown that IHDP outperforms DHP when it comes to online learning, accuracy and robustness. Furthermore in the case of rapid changing system dynamics (i.e. faults or damages) IHDP can estimate a new incremental model and a new near optimal control policy online without losing control. This is highly desirable for adaptive flight control. Furthermore IDHP has recently been demonstrated to work on a simulation model of the Cessna Citation II [32, 42, 45].

2.2.2. Safe Reinforcement Learning

Throughout history reinforcement learning has followed the principle of trial-and-error learning. In theory infinite exploration is required to reach the optimal policy. Finding a balance between exploration and exploiting the optimal policy, or the exploration-exploitation dilemma, is a key component of reinforcement learning. Too little exploration means the optimal policy is not found, whilst delaying exploitation for too long means more performance could have been achieved. In practice agents often spend more time exploring at the beginning of their training after which they exploit their found policy.

In real life situations exploration could even lead to dangerous situations [37, 39]. In a computer simulation of a chess game the worst possible outcome is a lost game. Aircraft exploring their entire state-space however could end up in an unrecoverable stall or dive. Robots could possibly damage themselves or their surroundings when exploring all possible states [1, 28, 30]. Therefore the concept of safety is of paramount importance when applying reinforcement learning on real-life problems.This section is not intended as a complete survey of safe reinforcement learning, rather it summarizes the research performed on safe reinforcement learning taken as inspiration for this thesis.

Reinforcement Learning primarily uses rewards to get information about the system. Actions leading to unsafe states will receive negative reinforcement in the form of lower rewards or penalties. In the long run these penalties will lead to the policy not visiting these states anymore. However if the rewards are not tuned properly or the occurrence of the state is low, the agent might not be able to learn that the state is unsafe. Furthermore visiting an unsafe state once might already do damage to the agent or the environment. This leads to the need of additional required methods in order to safely explore an environment without the risk of damaging oneself or its surroundings.

In this section methods for safe reinforcement learning are discussed. They are grouped in two categories:

- Constrained exploration methods, which constrain the policy of the actor to a safe policy
- Controller based methods, which combine non-reinforcement learning control methods with reinforcement learning methods to obtain a safe policy

Furthermore it is important to realize that reinforcement learning uses exploration to obtain knowledge. If this exploration is thus somehow limited external knowledge needs to be provided [23, 27]. In [23], which is widely accepted, different categories are used applicable to all reinforcement learning problems to differentiate between safe reinforcement methods, however for simplicity the methods found to be applicable to adaptive flight control in this study can be assigned to the categories listed above.

Constrained Exploration

In this research constrained exploration is used to describe reinforcement learning methods that constrain the outcome of the actor in some way to a set of safe actions. This means that only actions are taken that are known to keep the agent in a safe state. Figure 2.8 visualizes the concept using the definition of allowable policies to indicate the safe policies in comparison to the total policy space.

Actions that are not in the allowable policy may very well lead to a safe state, however since the agent is not aware of this, the action will not be taken. As exploration continues and the agent learns more about its environment it might increase the allowable policy space to include more actions that it now deems safe base on previous experience. In order for this type of method to succeed some initial knowledge about the agent and its environment are required [3, 5, 4].

Garcia and Fernandez [24] introduced the PI-SRL algorithm which uses a case-based risk function, based on the distance between an unknown and a known state, to asses the safety of unknown states during exploration. It showed its effectiveness on four different continuous problems, autonomous car parking, the cart-pole problem, helicopter hovering and business management. The authors here assume that a safe policy is already known, but increases performance is desired. As the agent moves away from its baseline policy it can increase its known state space. As the distance between known and unknown states increases so does the risk, based on a configurable risk-parameter new policies can be explored in a organized manner.

The authors underline the importance of an initial baseline policy, however poor it performs, as this directs the exploration (in comparison ϵ-greedy explore completely random). Although not entirely failure free the authors state that with PI-SRL higher rewards can be obtained with less failures in comparison to other risk-sensitive methods.

Another research presented by Mannucci et al. [48] introduces Safety Handling Exploration with Risk Perception Algortihm (SHERPA) and OptiSHERPA. SHERPA also includes risk in its decision process. Here it is assumed that the agent can perceive risk with onboard sensors. This enables an onboard safe-state-space model to be updated by the risk perceived in nearby states as it explores. Furthermore safe exploration is guaranteed by means of backups. This is described as a sequences of control inputs that bring the system close to a state it has visited in the past, and thus safe, when a state with high

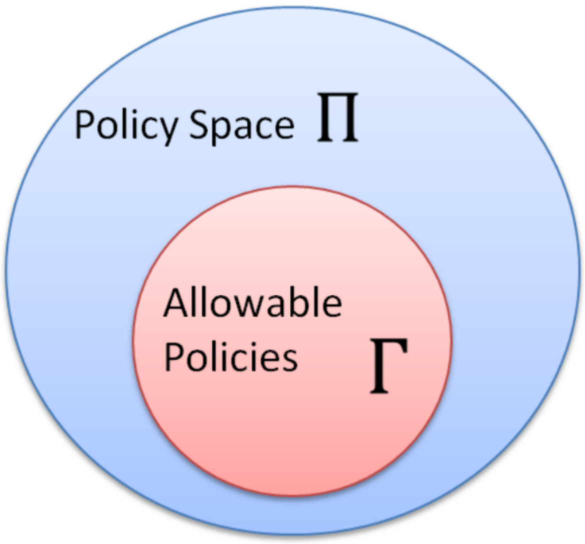

Figure 2.8: Visualization of the constrained or safe policies as a subset of the total policies.[23]

risk is encountered. Users will have to provide a baseline policy and an uncertain model of the system, a bounded model is used to take the uncertainty of the system into account. The procedure using backups is visualized in Figure 2.9.

OptiSHERPA adds metrics to assess safety for all available actions and includes an evasion strategy which relies on the current belief of the state space. SHERPA is demonstrated on a UAV exploring an unknown indoor environment, OptiSHERPA is demonstrated on simulation of a fighter aircraft exploring its flight envelope.

Next Dalal et al. [21] achive safety by using a safety layer located behind the agent. The role of the safety layer is to perturb the action given by the policy such that the system states stay within a set of upper bounding constraints. The result is a zero-constraint-violations method shown using a ball tracking task and a spaceship control task. The safety layer can be added to any existing learning algorithm. Since exploration is now constraint to safe states the addition of a safety layer also proved to increase convergence speed.

In order for this method to work the authors assume that the boundaries of the environment are known in advance. Secondly the dynamics of the system are first approximated with a linear model using a set of state-action tuples (s_j, a_j, s'_j). In this research the authors obtained these tuples by first exploring the environment using random exploration policy. Then with knowledge of the system already in the safety layer, training of the reinforcement learning agent could begin. During this stage the linearized model can continually be updated as new experience comes in.

As the agent is trained each action given by the policy is fed through the safety layer. Here Equation 2.24 is solved such that an action is selected as close to the original action as possible but still within the constrained space.

$$\begin{gathered}\arg\min_a \frac{1}{2}||a - \mu_\theta(s)||^2 \\ \text{s.t.} c_i(s,a) \leq C_i\end{gathered} \tag{2.24}$$

To minimize the computational cost the authors furthermore state that only one constraint is active at a time, or rather only one constraint can be crossed at a single time. This leads to minimal extra computational cost and a fully differentiable solution.

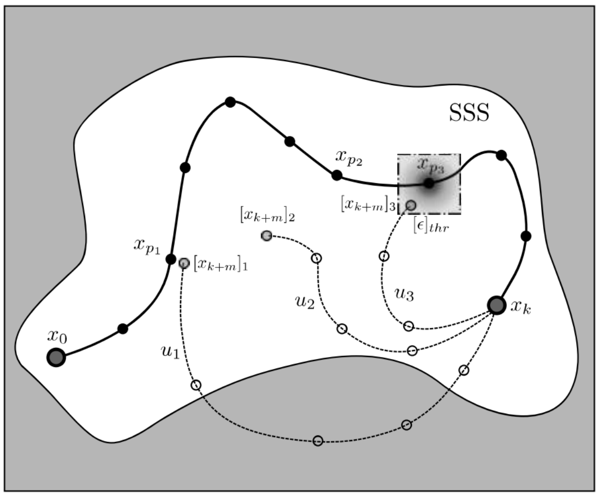

Figure 2.9: Example of backup generation. As an agent starting from $x(0)$ is now reached x_k following the trajectory outlined. The dashed lines indicate the means of the proposed backups. Backup u_1 is discarded as it violates the known safe state space, u_2 is also discarded as $[x_{k+m}]_2$ does not satisfy the closeness condition with an earlier explored state. Backup u_3 is completely in the safe state space and satisfies the closeness condition.[48]

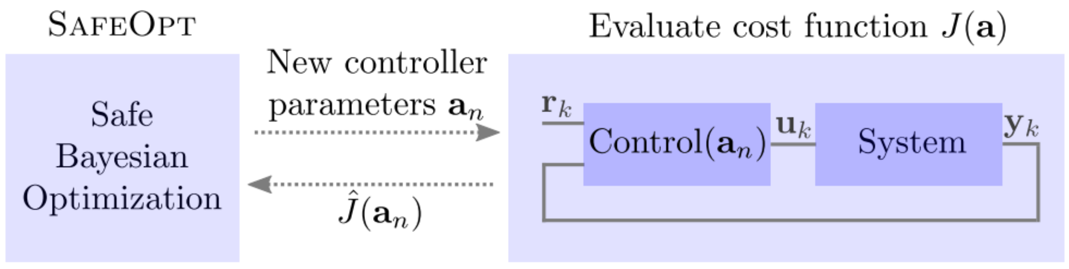
SafeOpt
Safe Bayesian Optimization
New controller parameters $\mathbf{a}_n$
$\hat{J}(\mathbf{a}_n)$
Evaluate cost function $J(\mathbf{a})$
$\mathbf{r}_k$
Control($\mathbf{a}_n$)
$\mathbf{u}_k$
System
$\mathbf{y}_k$

Figure 2.10: Overview of the SafeOpt method. Controller parameters are updates based on a safe bayesian optimization. [14]

Controller based reinforcement learning

As opposed to constrained exploration, controller based reinforcement learning makes use of analytical controllers to control the system at hand. Reinforcement learning is then used to increase performance in the safe but often very conservative controllers [11, 13, 15]. This can be done in different ways, some methods found to be particularly interesting for adaptive flight control are highlighted below. These type of methods require an initial safe controller as a starting point for their exploration.

SafeOpt as presented by Berkenkamp, Schoellig, and Krause [14] is a safe optimization algorithm making use of Bayesian optimization. This is particularly useful as Bayesian optimization tries to find the global maximum of an unknown function where it ”assumes that evaluating the function is expensive, whilst computational resources are cheap” [14]. This fits nicely in the field of safe reinforcement learning where evaluating a policy on a real system takes time, causes wear, and has the probability of leaving the safe boundaries. In order to do so a Gaussian Process (GP) is used to find an approximation of a non-linear action to value map.

Figure 2.10 shows an overview of the SafeOpt method. An initial safe controller is used control the system. Based on the returning values, SafeOpt adaptively selects new controller parameters that increase performance. In order to ensure that the new controller is also safe, SafeOpt restricts itself to only learning a safe set of controller parameters by enforcing Equation 2.25 which states that only parameters that lead to an improvement in comparison to a minimum threshold value $J_{\min}$.

$$S = \{a \in A \mid J(a) \geq J_{\min}\} \tag{2.25}$$

As learning starts most of the value function is uncertain and thus only parameters with a high probability of being safe are used. SafeOpt then tries to find the global maximum of the current known set of

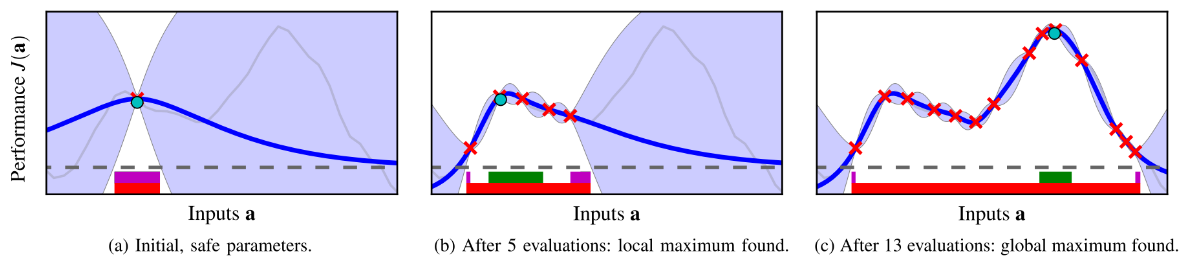

Figure 2.11: Optimization with the SafeOpt algorithm. The blue line indicates the approximation of the underlying function, with a confidence interval (light blue). The grey dashed line represents the minimum value J_{min} required to guarantee safety. Through samples the safe region is expanded and the global optimum is found.[14]

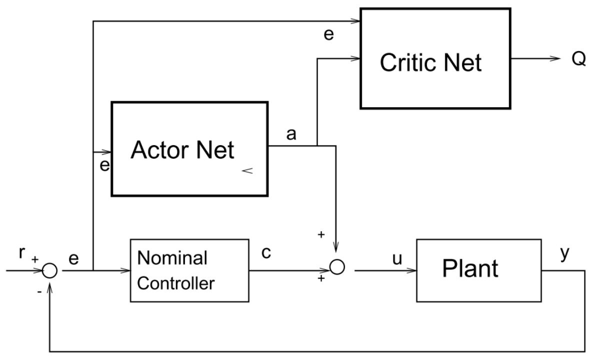

Figure 2.12: Actor-Critic learning in conjunction with a robust controller that ensures stability.[41]

parameters called potential maximizers, additionally it also aims to expand its safe set of controllers through potential expanders. An example of how SafeOpt works is visualized in Figure 2.11.

Furthermore the authors show how SafeOpt is used to fly a quadrotor and increase its performance. Continuous development also led to a further decoupling of safety and performance to allow for application on robotics where high performance through high gains can have a negative effect on safety [12]. Junell et al. [36] conducted a similar test with similar results.

Next the method shown by Kretchmar et al. [41] is discussed. Here reinforcement learning is applied in conjunction with a robust controller. Robust control is able to deal with uncertainties in a model. It allows for a stable controller by sacrificing performance to gain stability. This is particularly helpful when the model used to create the controller is not an accurate representation of the reality. This is done by creating an uncertainty bound. In this research the authors use this attribute of robust control to counter uncertain outputs of a reinforcement learning controller.

An overview of the method is shown in Figure 2.12, where the nominal controller is a robust controller. Based on the known uncertain plant dynamics and a known maximal disturbance of the actor network a robust controller is generated that creates a stable system. As can be seen an actor-critic learning method is used. During learning the actor will learn what actions to add to the robust controller in order to increase performance based on feedback provided by the critic. Therefore an optimal policy can be created in reality even though no accurate model of the system is available. The weights of the neural network are bounded to ensure that the total system is always stable. If an update leads to an increase of the weight of the neural network outside this bound, the robust controller is retuned to allow for different disturbances to be added by the actor. The disturbance magnitude is calculated by converting the neural network to a linear time invariant uncertainty model. The result is a stable system even when the actor network is being trained.

Figure 2.13, Figure 2.14 and Figure 2.15 show the different responses to a step input for a conventional controller, a robust controller, and a robust controller aided by an actor-critic respectively when tuned on an uncertain distillation column model and then used on the real system. It can be seen that the conventional controller has a high peak due to the mismatch in model and reality, the robust controller is stable but slow, and the solution as presented by Kretchmar et al. [41] performs best. Recently

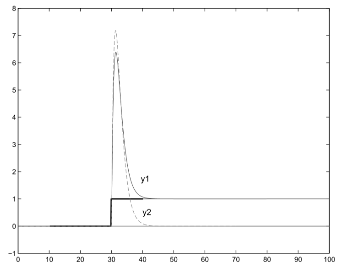

Figure 2.13: Response to a step input of a conventional controller tuned on a model and evaluated on the real system.[41]

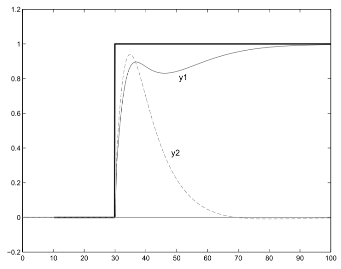

Figure 2.14: Response to a step input of a robust controller tuned on a model and evaluated on the real system.[41]

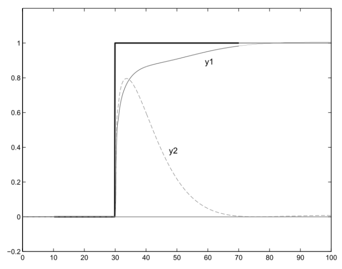

Figure 2.15: Response to a step input of a robust controller in conjunction with an actor-critic on the real system.[41]

Jin and Lavaei [35] used a similar approach on a multi-agent system.

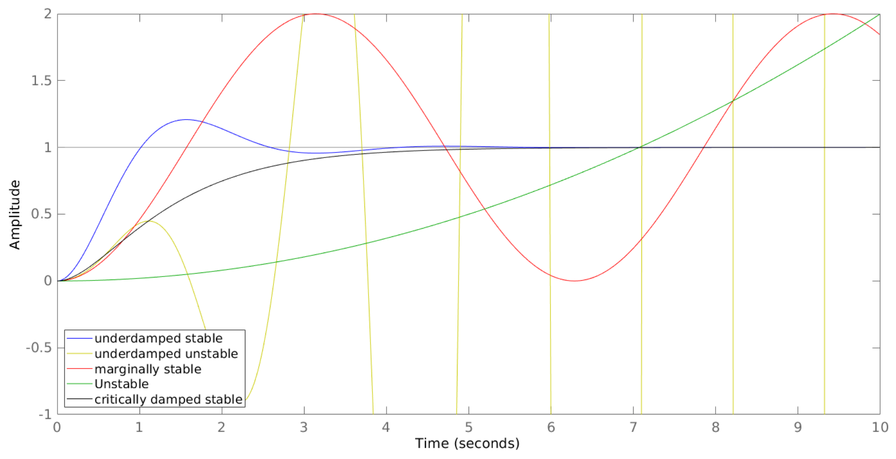

Figure 2.16: Overview of different dynamics in a system response to a step input.

2.3. Safety in flight control

As in this research safe reinforcement learning is applied on adaptive flight control in this section safety in flight control is discussed. From a control engineering perspective the stability of the entire aircraft system is key to ensure a safe flight. Other engineering departments will also pose limits on the aircraft leading to the creation of a safe flight envelope. The types of limits encountered can given an indication as to which safe reinforcement learning method is optimal for adaptive flight control. This section is divided into two sections namely stability and safe flight envelope.

2.3.1. Stability

Stability in a system is key not just for aircraft control, but for all systems. Through properly tuned controllers unstable systems can be made stable. On the other hand an ill-tuned controller can decrease stability. A system is defined as stable when it always returns near the steady state. An unstable system moves further away from this state without being bounded when an input is given. Lastly we speak of marginal stability when a system is somewhere between stable and unstable. Furthermore a system can be underdamped, critically damped or overdamped, which characterizes the oscillations experienced by the system. Figure 2.16 shows all these types of stability through a system response when excited by a step input.

For an aircraft it is desirable to attain a stable system. This will contribute to safety when the aircraft is exposed to disturbances such as wind gusts or turbulence, as the aircraft will always return to its steady state. This eases the workload of the pilot. Furthermore as heavily damped systems also create a slow response, aircraft are often slightly underdamped to increase performance [19]. If an aircraft initially does not display these control characteristics the controller needs to be tuned in order to attain these.

When using PID controllers the gain and phase margin can be consulted to obtain some sort of safety margin. For a neural network and adaptive flight controllers in general however this is not the case. Bhattacharyya et al. [16] present a complete overview of the challenges that come when certifying adaptive controllers. The authors also state that when the adaptive controllers provide a significant benefit whilst dependability can be shown, it is worth considering how this controller can be certified. The certification process is outside the scope of this research. Stepanyan et al. [62] introduce performance metrics which could possibly be used to certify adaptive controllers. Here the authors state that, just as with PID controllers, the gain margin can still be used when assessing the stability of an adaptive flight controller. Furthermore the phase margin can only be used to check for stability if the adaptive flight controller has converged to a steady solution. Next they introduce the time-delay margin as an extra measurement which is defined as the time delay t_d the closed loop system can tolerate without instability.

For the transient response metrics such as overshoot or undershoot are used to indicate performance of the controller. For adaptive controllers however the authors of [62] propose to use a transient performance measurement, TPM, as shown in Equation 2.26. Here $x(t)$ indicates the state of the adaptive system and $x_m(t)$ is the reference given. It can be interpreted as the maximum occurring error between t_0 and the final measurement t_f. The measurement is normalized to allow for a general comparison. An alternative would be to use the $\mathcal{L}_2$ norm instead of the $\mathcal{L}_\infty$ norm which allows for intervals to be analyzed and takes oscillations into account. This is shown in Equation 2.27.

$$TPM_\infty = \frac{||x(t) - x_m(t)||_{\mathcal{L}_\infty}}{||x_m(t)||_{\mathcal{L}_\infty}} = \frac{\max_{t\in[t_0,t_f]} ||x(t) - x_m(t)||}{\max_{t\in[t_0,t_f]} ||x_m(t)||} \tag{2.26}$$

$$TPM_2 = \frac{||x(t) - x_m(t)||_{\mathcal{L}_2}}{||x_m(t)||_{\mathcal{L}_2}} = \frac{\sqrt{\int_{t_0}^{t_f} ||x(t) - x_m(t)||^2 dt}}{\sqrt{\int_{t_0}^{t_f} ||x_m(t)||^2 dt}} \tag{2.27}$$

Adaptive flight controllers should be able to quickly learn a new policy in order to create a stable system with the desired performance when a failure or damage occurs. This requires online learning in combination with model-free reinforcement learning as shown by Prokhorov and Wunsch [54] and Zhou, Kampen, and Chu [76]. Also the safety constraint as shown by Jin and Lavaei [35] could assist in providing stability guarantees. The metrics introduced above will still work when adaptation to a changed system is tested.

2.3.2. Safe Flight envelope

The safe flight envelope of an aircraft indicates in what states the aircraft can fly as directed by the design limits. This is entirely different from control stability in the sense that a controller can be used to change the type of stability of the aircraft, but the design limits can not be altered. These limits can exist based on all types of constraints. For example aerodynamic limits can lead to a maximum angle of attack, structural constraints have an influence on the loadfactor, and the engine thrust will affect the maximum altitude that can be reached. The current state of the aircraft also has an influence on the actual limit, e.g. the current aircraft weight affects the stall speed.

All these limits combined generate a so-called safe flight envelope. An example of such an envelope can be seen in Figure 2.17[1]. Here it can be seen that how the maximum load factor is affected by the airspeed of the aircraft. Safe operation can be guaranteed as long as the aircraft stays in the grey area. Similar plots could be drawn also showing altitude. If an aircraft finds itself outside this safe flight envelope extensive studies into upset recovery strategies have led to an optimal return back to the safe flight envelope [20].

For regular airplanes these limits are usually known, this helps in designing a controller for the aircraft that keeps the aircraft within the safe operating regime. Airbus for example has a stall protection system which does not let the pilot pull the aircraft past the maximum angle of attack or load factor [17]. Knowing these limits can also help in creating a safe reinforcement learning controller, in subsection 2.2.2 the category of constrained exploration can easily cope with limits known beforehand.

Smaller aircraft or UAV's may not always have a completely known safe flight envelope. Here it becomes more difficult to pose hard limits on the reinforcement learning controller and risk based methods based on sensor measurements might be more appropriate as shown in [48].

If for some reason the dynamics of the system change, e.g. failures or damages occur, the safe flight envelope might also change. This is illustrated in Figure 2.18[2]. Although flight envelope prediction is outside the scope of this research, since the flight controller makes use of reinforcement learning it could potentially learn the new safe flight envelope and keep the aircraft in a safe state through the feedback it receives. Uncertainty bounds as used in [41] or risk signals used in [24] and [48] can help the controller obtain a safe operating region.

[1]http://aviation_dictionary.enacademic.com/879/basic_flight_envelope
[2]http://cs.lr.tudelft.nl/syscon/projects/flight-envelope-prediction/

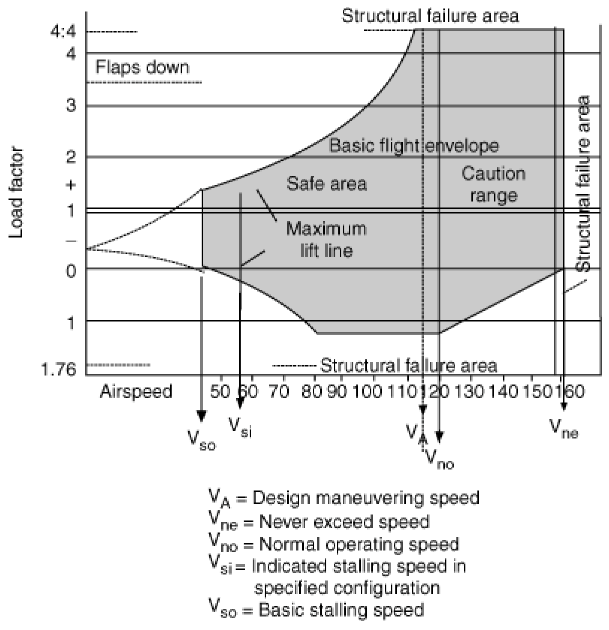

Figure 2.17: Example of flight envelope. (source: `http://aviation_dictionary.enacademic.com/879/basic_flight_envelope`)

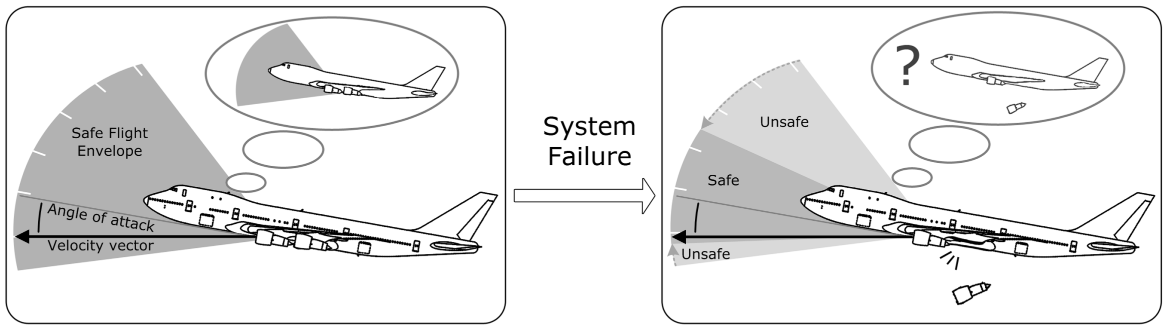

Figure 2.18: Example of adjustable flight envelope due to damage. (source: `http://cs.lr.tudelft.nl/syscon/projects/flight-envelope-prediction/`)

2.4. Conclusion

This chapter provides an introduction to reinforcement learning by first providing a historical background and then presenting the methodology of reinforcement learning. Also state-of-the-art research is discussed followed by an overview of safe reinforcement learning methods applicable to flight control. Lastly safety in flight control is discussed. The goal of the chapter is to get acquainted with the topic of safe reinforcement learning and create ideas in how to apply safe reinforcement learning to flight control problems.

In subsection 2.1.2 the basics of reinforcement learning are explained using discrete spaces later generalizing to continuous action spaces. From this we can conclude that reinforcement learning in its purest form will converge to a solution by seeing each state-action pair infinite times. This will build an accurate value function for each discrete state-action pair. TD-learning has shown to outperform MC learning in terms of convergence speed, as approximations are used to build the estimate and training can be performed online. Online training is also a requirement for this research. In order to move to continuous spaces function approximators are introduced. They are able to generalize to unseen states and thus inherently capable of applying reinforcement learning in continuous space.

State-of-the-art methods found in section 2.2 show that reinforcement learning is capable of solving complex tasks with the use of ANN. Following this example ANN are also used as approximators in this research. However learning without failing is not trivial, and not a direct objective of deep reinforcement learning. Methods using some sort of model such as DHP and IHDP are able to converge to solution much faster due to the extra knowledge provided. Even with this extra knowledge provided an offline training period may be required before the actual training begins. Furthermore constrained exploration allows for incorporation of knowledge from the environment and could be beneficial to creating a safe reinforcement learning algorithm. Controller based methods show promising results, but require an initial safe controller as a baseline, something which may not be present for all problems.

Lastly section 2.3 shows us the type of limits an aircraft has to adhere to in order to stay in the safe space. It also introduces the TPM_2 and TPM_∞ metric which can be used on adaptive controllers to test the stability of the controller.

3 Preliminary Analysis

In the literature study it was found that using constrained exploration offers possible solutions in the field of adaptive flight control. The safe flight envelope of the aircraft can be taken as the exploration space of the reinforcement learning agent. In this chapter a preliminary analysis is performed in order to validate that constrained exploration works in adaptive flight control. Furthermore it is investigated whether the safe flight envelope can be taken as a safe exploration region. Lastly the analysis is used to gain experience in working with safe reinforcement learning.

First the simulated environment that will be used to test the performance is explained in section 3.1. Next the applied method is presented in section 3.2. This is followed by a discussion of the results in section 3.3.

3.1. Problem statement

In order to test the performance an environment is created that simulates an aircraft tracking a height reference signal. The environment is based on the OpenAI gym structure [18]. This means that the environment takes an action as input and returns the next observation, the reward and whether the episode is terminated. Figure 3.1 shows a sample of screenshot of the environment. As it is only a very rudimentary simulation focused primarily on safety the aircraft dynamics are simplified to Equation 3.1. To also asses the adaptiveness of the different algorithms after a set number of episodes Equation 3.2 is used where the action a now leads to an opposite result than before. The rewards are given according to Equation 3.3, here h_{r_t} and h_{s_t} are the reference height and the aircraft height at timestep t respectively. ΔH represents the safety bound used for the safe flight envelope and is used to obtain a normalized value. Should the episode terminate a reward of -50 is given. Lastly Equation 3.4 shows how the reference signal is created. The constants A_1, A_2, A_3, f_1, f_2 and f_3 are randomly generated at the start of each simulation.

$$\begin{bmatrix} u \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot a \tag{3.1}$$

$$\begin{bmatrix} u \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} \cdot a \tag{3.2}$$

$$R_t = \begin{cases} 1 - \frac{|h_{r_t} - h_{s_t}|}{\Delta H}, & \text{if} |h_{s_t}| \in \Delta H \\ -50, & \text{otherwise} \end{cases} \tag{3.3}$$

$$h_{r_t} = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + A_3 \sin(2\pi f_3 t) \tag{3.4}$$

To further simplify the problem the environment is discretized such that function approximators are not yet required and the focus can solely lie on the safety aspect of the research. To asses the safety a limit on the minimum and maximum height is set, with respect to the starting altitude, as can also be seen in Figure 3.1. For this research $\Delta H = 5$. A second limit is imposed when the aircraft obtains a vertical velocity of more than two height steps per timestep. These limits are used to represent the safe

Table 3.1: Overview of the inputs and outputs used by the environment and consequently the agent.

	Environment	**Agent**
input	$a(-1,0,1$	$h_{r_t} - h_{s-t}, \quad u$
output	$h_{r_t} - h_{s_t}, \quad u$	$a(-1,0,1)$

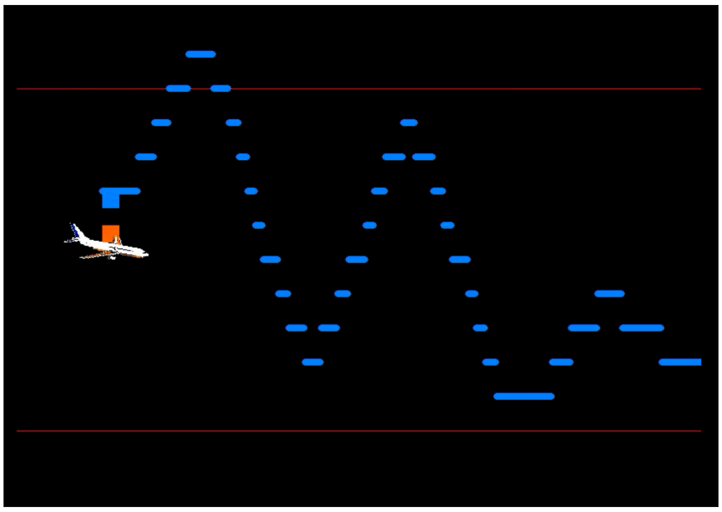

Figure 3.1: Discrete training environment used to test the performance of the agents. The red lines indicate the boundaries of the safe flight envelope, the blue line is the trajectory that needs to be tracked.

flight envelope of the aircraft and moving outside of this region results in a termination of the episode and a negative reward bonus. To challenge the controller even further the reference signal is allowed to move outside the safe flight envelope, the controller should then decide to not follow the reference signal due to safety limits. Table 3.1 gives an overview of the states outputted and the inputs taken by the environment. It also shows that the agent will have to provide exactly the opposite, it should take the outputs of the environment as input and provide the action to take as output.

3.2. Methods

In order to gain insight in constrained exploration methods it is decided to implement the methods shown in [21] also explained in subsection 2.2.2. This method is chosen based on the promising results shown in the research and due to the fact that it can be applied to an already existing reinforcement learning framework which simplifies implementation. Tabular Q-learning is chosen as reinforcement learning framework as this is proven to work well with discretized environments and is also easy to implement in an online learning fashion.

3.2.1. Tabular Q-learning

In subsection 2.1.2 the Q-learning algorithm was introduced as a form of TD-learning. As such it makes use of older estimates in order to update its current estimates. Equation 3.5 shows again how the Q-value of each state action pair is obtained. For this analysis $\alpha = 0.2$ and $\gamma = 0.8$.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{3.5}$$

In order to apply tabular Q-learning a table, or array, needs to be generated which has a separate entry for each state-action pair encountered in the learning process. Next the state from the environment is received and the table is used to find the best action for that state, in order to facilitate exploration the action is selected based on a probability distribution provided by the softmax of the action values. This action is then applied and the reward and next state are received. First the reward and the new state are used to update the value estimate of the previous timestep, after which the new state is used to again select the best action. In Algorithm 1 the complete pseudocode can be found.

Algorithm 1 Q-learning applied to test environment

```
Require: $N$, $\alpha, \gamma$, env
  initialize ($Q[s,a]$ for all $s \in S, a \in A$) = 0, $\mathrm{ep}_n = 0$, $R_{\mathrm{array}}$
  while $\mathrm{ep}_n \leq N$ do
    $s$, done ← env.reset
    while done = False do
      $a \leftarrow$ from $Q[s,:]$ using $\pi(a|s) = \mathrm{softmax} Q[s,a]$
      $s', r$, done ← using $a$ on env.step
      append $s'$, $r$, $a$ to $S_{\mathrm{array}}$, $R_{\mathrm{array}}$, $A_{\mathrm{array}}$
      $Q[s,a] \leftarrow Q[s,a] + \alpha\left[r + \gamma \max_a Q[s',a] - Q[s,a]\right]$
      $s \leftarrow s'$
      $\mathrm{ep}_n \leftarrow \mathrm{ep}_n + 1$
    end while
  end while
```

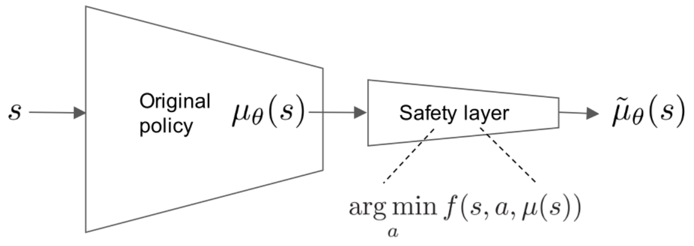

Figure 3.2: The safety layer is positioned behind the tabular Q-learning algorithm. Actions leading to unsafe states are minimally altered in order to stay in safe space.

3.2.2. Safety Layer

The purpose of the safety layer is first to guarantee that the action will lead to a safe state, and secondly to minimize the difference between the proposed action by the Q-learning algorithm and the safe action. This is shown in Equation 3.6.

$$\begin{aligned} &\arg\min_a \frac{1}{2}||a - \mu_\theta(s)||^2 \\ &\text{s.t.} c_i(s,a) \leq C_i \end{aligned} \tag{3.6}$$

For this purpose the safety layer, SL, is positioned behind the Q-learning algorithm. This is displayed in Figure 3.2. In [21], first an initial training phase is required in order to find the safe space. The authors then state that learning the safe space can continue during training of the reinforcement learning algorithm, but this did not provide them with any additional benefits. As this method is now applied on a flight control problem, more knowledge about the environment is already known. Whilst Dalal used 1000 random episodes with random actions in order to find the complete safe space, here only a few timesteps are required to find the state-space model using a rolling-window-least-squares method. The actual safety limits are assumed given as the safe flight envelope is usually known as stated in section 2.3. Furthermore as also the adaptiveness of the method is tested now the continuous training of the safe space is expected to benefit the performance.

In the safety layer the proposed action is used as input on the learned state-space system. The output is then used to check the margins with respect to the boundaries. If a proposed action leads to crossing the boundaries or a trajectory from which the boundaries are inevitably crossed, a different action is selected. This is done by calculating the action closest to the proposed action but still staying the safe space. The complete pseudocode for this problem can be found in Algorithm 2.

Algorithm 2 Q-learning followed by a safety layer.

```
Require: $N$, $\alpha, \gamma, \Delta H$, env
  initialize $(Q[s,a]$ for all $s \in S, a \in A) = 0$, $\mathrm{ep}_n = 0$, $R_{\mathrm{array}}$, $S_{\mathrm{array}}$, $A_{\mathrm{array}}$
  while $\mathrm{ep}_n \leq N$ do
    $s$, done ← env.reset
    $SL$ ← from exploratory actions taken in env and $\Delta H$
    while done = False do
      $a$ ← from $Q[s, :]$ using $\pi(a|s) = \mathrm{softmax} Q[s,a]$
      $a$ ← check (and alter if required) $a$ using $SL$
      $s', r$, done ← using $a$ on env.step
      append $s'$, $r$, $a$ to $S_{\mathrm{array}}$, $R_{\mathrm{array}}$, $A_{\mathrm{array}}$
      $SL$ ← check (and alter if required) using rolling window LS and $S_{\mathrm{array}}$, $A_{\mathrm{array}}$
      $Q[s,a] \leftarrow Q[s,a] + \alpha\left[r + \gamma \max_a Q[s',a] - Q[s,a]\right]$
      $s \leftarrow s'$
      $\mathrm{ep}_n \leftarrow \mathrm{ep}_n + 1$
    end while
  end while
```

3.3. Results

The Q-learning method and the Q-learning followed by a safety layer are both trained on the tracking task for 500 episodes with a maximum length of 100 timesteps. Furthermore after 250 episodes the internal dynamics of the aircraft changed from using Equation 3.1 to Equation 3.2 to test the robustness of the different methods to altering aircraft dynamics. As a reference a tuned PID controller has also been simulated. In this section the performance and safety of the algorithms are compared and discussed. First the performance is discussed more from a reinforcement learning perspective. This is followed by a discussion from a safe flight control perspective.

3.3.1. Performance

In order to measure the performance of the different methods use is made of the total rewards received per episode. This can be seen in Figure 3.3 for all three methods. Furthermore in Figure 3.4 the length per episode is presented.

When looking at Figure 3.4 it can be seen that adding a safety layer worked as expected. No premature episode terminations are required and using the rolling-window-least-squares method quickly provided enough information about the system. Q-learning without a safety layer proves capable of learning the task, but fails prematurely many times in the process. From Figure 3.3 it can be seen that Q-learning requires the negative reinforcement through exploration in order to discover the safe space. Adding a safety layer also improves the convergence speed of the method. This can be explained by the fact that since episodes last longer more rewards are received and more learning takes place. It can be seen that both Q-learning methods cannot outperform a tuned PID controller. From episode one the PID controller is able to perform well receiving almost maximum rewards, it only failed when the reference trajectory itself went outside the safe region. However as the dynamics of the aircraft change the PID controller is no longer of use and both Q-learning methods are able to adapt and perform. It takes both Q-learning methods around 75-100 episodes to adapt to the new dynamics and perform around the same level as before. Interestingly the Q-learning with safety layer starts performing better again after the change in aircraft dynamics but never reaches the same performance as before. Without safety layer however the performance continues to increase. This could be explained by the fact that softmax was used to create a categorical probability function. When the state action values are close together there is no action that is clearly better. Unlearning a value is harder when a safety layer is active as no negative reinforcement is applied. When the aircraft dynamics change, the rolling-window-least-squares method is again able to quickly provide enough information such that the aircraft stays within the safe space.

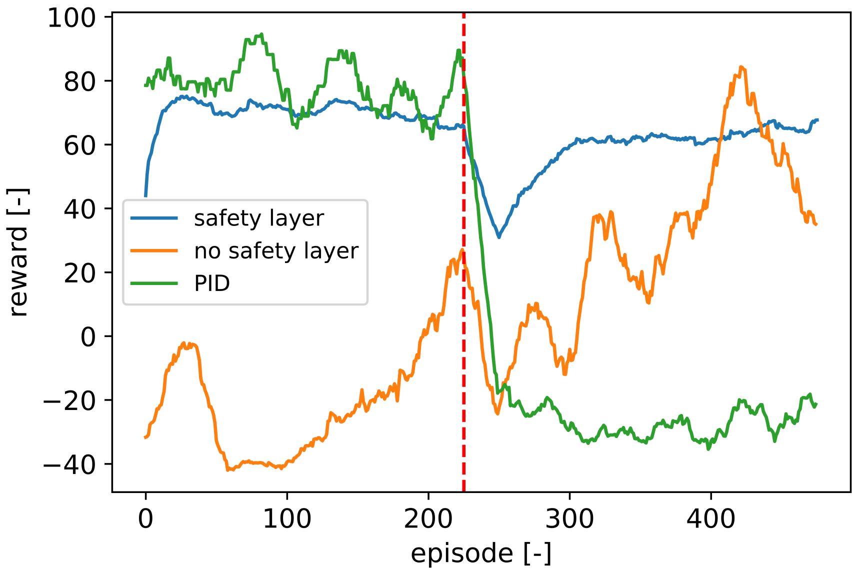

Figure 3.3: Average reward per episode for the method with and without safety layer. The dashed red line indicates where the dynamics of the aircraft changed. Also a PID controller is added for comparison

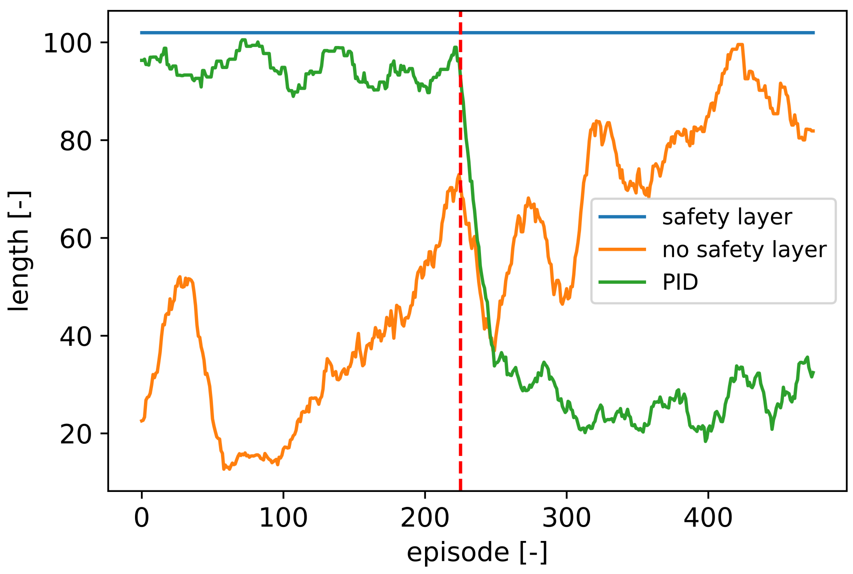

Figure 3.4: Average length per episode for the method with and without safety layer. The dashed red line indicates where the dynamics of the aircraft changed.

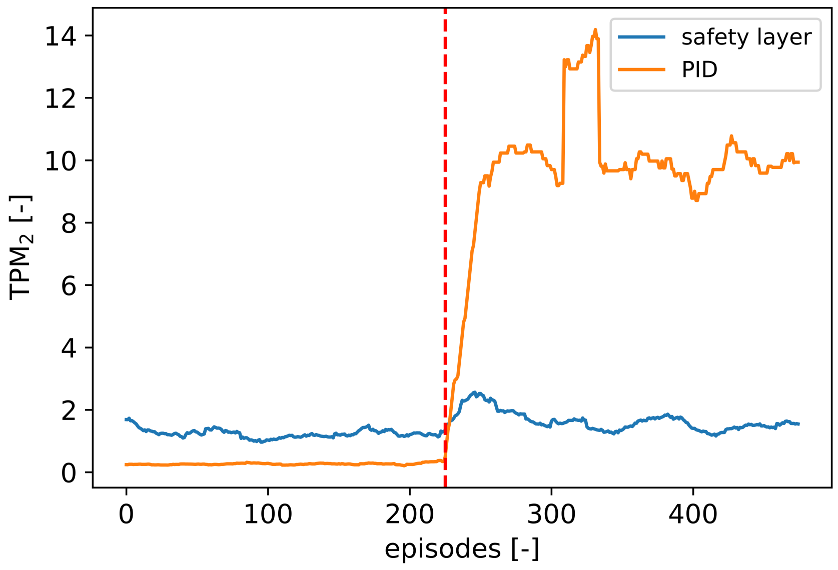

Figure 3.5: TPM_2 for a PID controller and the Q-learning method with safety layer. The red dashed line indicates where the dynamics of the aircraft changed.

Table 3.2: Overview of TPM_2 and TPM_∞ for each method using only the episodes before the internal dynamics changed.

	Q-learning	Q-learning with safety layer	PID-controller
TPM_2	27.4	1.28	0.29
TPM_∞	3.98	0.92	0.14

3.3.2. Safety

In order to inspect the safety of the different methods use is made of Figure 3.4, where the length of each episode is shown, Figure 3.5 which presents the transient performance measurements TPM_2, and Table 3.2, which shows the TPM_2 and TPM_∞ values calculated using only the episodes before the change in aircraft dynamics, this allows for a reliable benchmark from the PID controller.

As Q-learning with safety layer is the only method to consistently reach the maximum number of timesteps the safety layer enables the aircraft to successfully stay within its safe flight envelope. Furthermore using constrained exploration showed capable of dealing with adaptive flight control as expected. Even when the reference trajectory moves outside of the safe flight envelope the aircraft is able to stay within the safe flight envelope.

The stability performance is not satisfactory compared to the PID controller. The TPM_2 of the Q-learning method with safety layer reduces slightly from the start but stays at 1.28 compared to 0.29 for the PID controller. This means that a lot of oscillation is present throughout the flight, this was also seen in visualisations of the simulations. Oscillations are not only uncomfortable for the passengers, but could also lead to excessive accelerations and loads. The TPM_∞ indicating the maximum occurring error is also larger than the PID controller, but still within the safe flight envelope. It can be seen that adding a safety layer increases the stability of the aircraft, but does not yield significant performance to allow for comfortable and safe flight.

As only a discretized tabular Q-learning algorithm was used it was not expected that the method would create competitive performance to a PID controller. The stability might improve when using a more sophisticated learning framework such as an adaptive critic design using function approximators. Secondly using controller based reinforcement learning could also create significant improvement to the stability of the system.

3.4. Conclusion

In this chapter a constrained exploration method, described in [21], in combination with tabular Q-learning on an discrete control problem calling for an adaptive flight controller is successfully implemented. It is seen that the aircraft does not leave the safe space when the safety layer is active. Furthermore a change in dynamics is recognized such that the safety layer continues to work. It can be seen that the rewards received per episode are comparable to a PID controller. When looking at the stability however the safety layer provides some help, but not enough. The aircraft is constantly oscillating and performs considerably worse than a tuned PID controller. The use of more advanced continuous reinforcement learning algorithms such as DHP or IHDP can improve the stability. Furthermore ideas from controller based reinforcement learning could also benefit this issue.

III

Additional Results

4 Example runs for experiment 2: height tracking task

In order to provide further understanding about the failures encountered in experiment 2 additional figures showing individual runs can be found here. In Part I the general results are presented by using all of the 100 runs for each data point. As such the total failures and the average reward was shown. Furthermore, an individual run was shown for the low learning rates to further show the impact of SIDHP at low learning rates. In this chapter figures are shown for the optimal learning rate, which was discovered to be 1, and a run using a learning rate of 5 is displayed.

First, in Figure 4.1 and Figure 4.2 the height and elevator deflection respectively, for IDHP and SIDHP, are shown when the ideal hyper parameters are used. It can be seen that in general the lines overlap, and follow the height reference accurately. Within the first seconds IDHP and SIDHP both know nothing about the aircraft and thus first deviate only to learn the correct control policy. During this period the safety layer is activated briefly which results in slightly less aggressive commands as can be seen in Figure 4.2. Since the safety layer makes use of the incremental model, which converges faster than the actor, it is able to propose safe actions even before the controller converges to a suitable control policy. Therefore it increase safety during exploration.

Next, in Figure 4.3 and Figure 4.4 the height and elevator deflection are shown when a learning rate of 5 is used. Other hyperparameters have not been altered. Again an initial exploration period can be seen after which both IDHP and SIDHP converge. As the learning rate is now higher learning is more aggressive. In Figure 4.4 this can be seen as during exploration the maximum elevator deflection is used for around 5 seconds on IDHP. Using SIDHP the safety layer is able to avoid such rapid changes. For the displayed runs it can be seen how, although at a different time, both IDHP and SIDHP adapt their policy too aggresively due to the high learning rate. IDHP loses track of the control policy at around 110 seconds which again results in maximum elevator deflection being applied for a long period of time untill the simulaton crashed due to numerical errors. SIDHP also shows a similar behaviour around 120 seconds, but the safety layer helps in converging back to the control policy. It should be noted that the safety layer is not always able to guarantee safety as seen in Part I. If SIDHP also fails a behaviour similar to IDHP is seen.

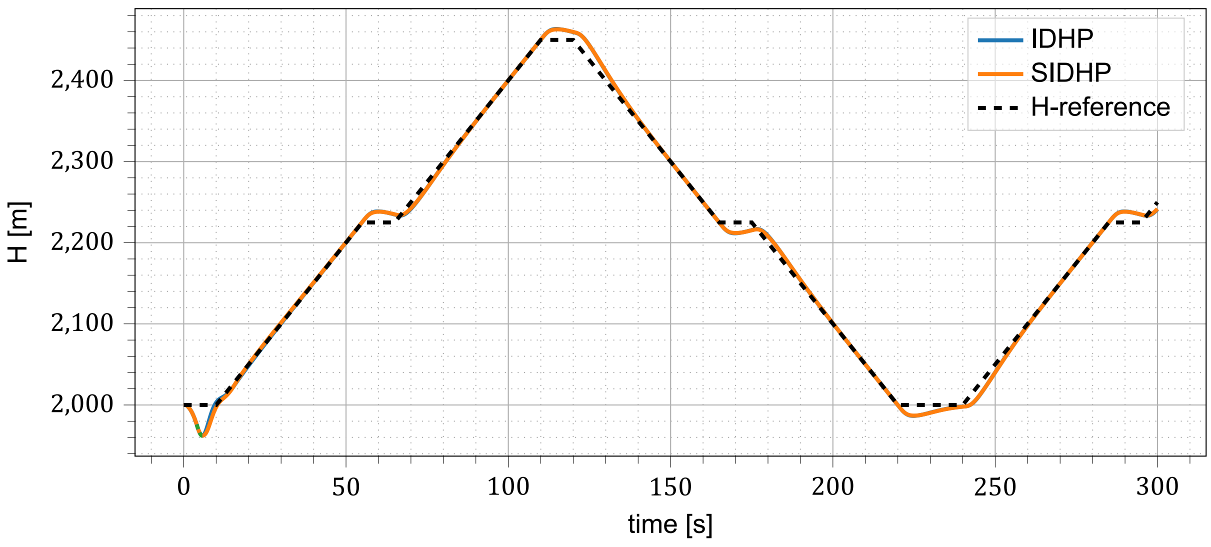

Figure 4.1: Experiment 2: results for IDHP and SIDHP using ideal hyper parameter settings for a single run.

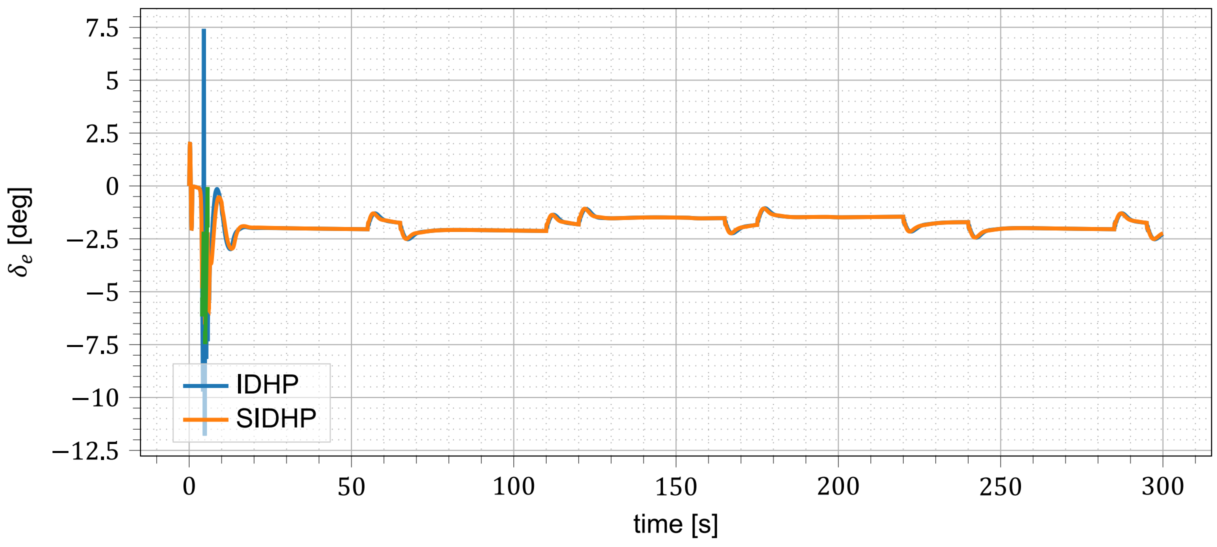

Figure 4.2: Experiment 2: elevator deflection of IDHP and SIDHP when using ideal hyper parameter settings.

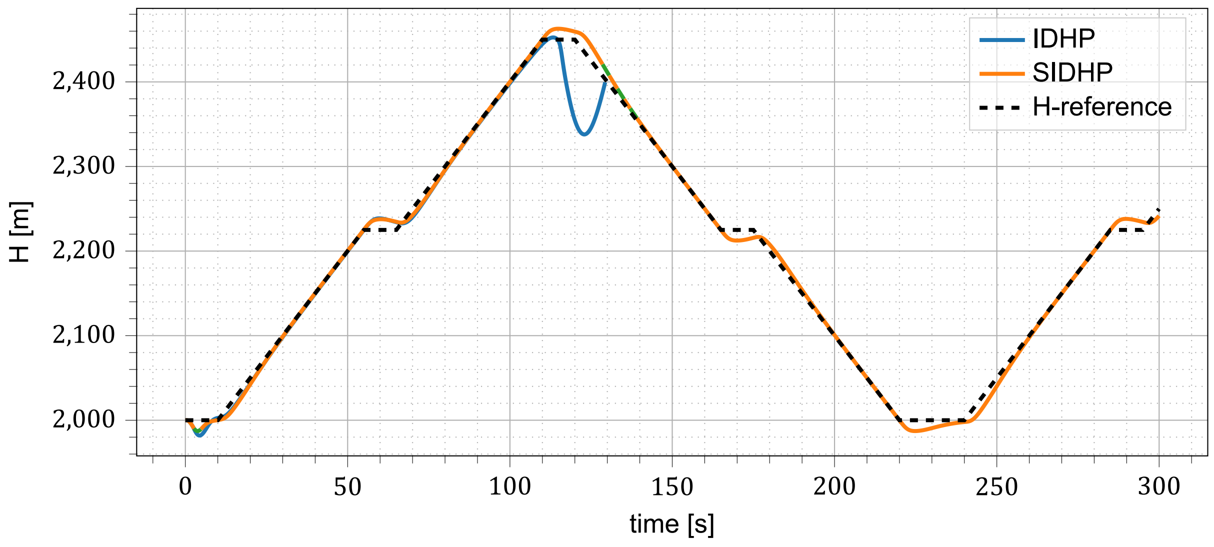

Figure 4.3: Experiment 2: results for an unsuccesful IDHP run and a succesful SIDHP run when using a learning rate of 5. An exploding gradient results in a failed run. SIDHP is able to control the exploding gradient. Green indicates the safety layer is active

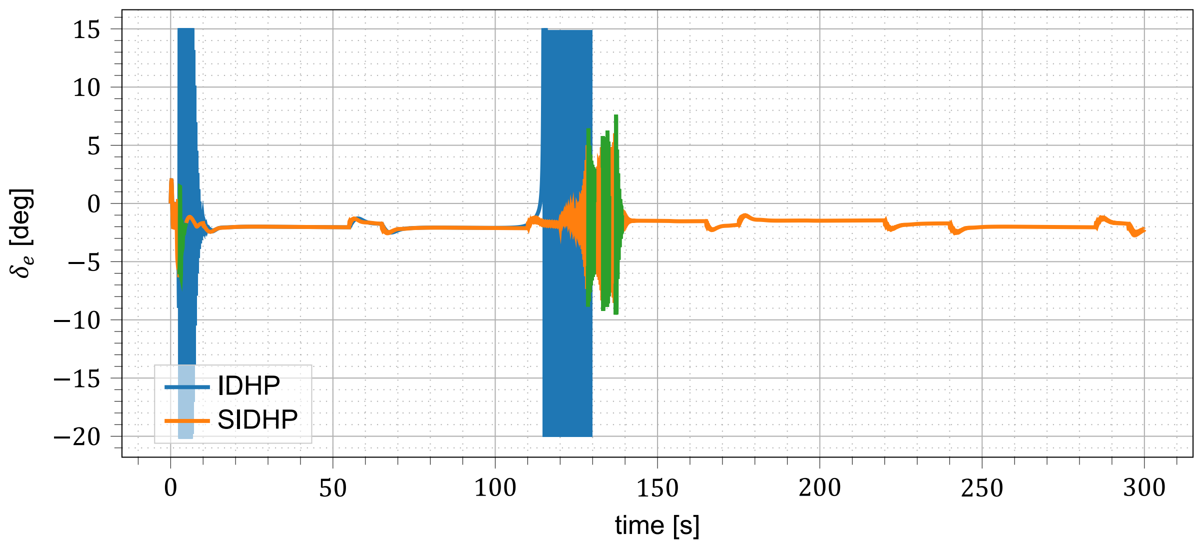

Figure 4.4: Experiment 2: elevator deflection for an unsuccesful IDHP run and a succesful SIDHP run when using a learning rate of 5. An exploding gradient results in a failed run, as can be seen in the large deflection. SIDHP is able to control the exploding gradient. Green indicates the safety layer is active.

5 State Predictions

SIDHP makes use of an incremental model to predict future states. In this chapter the usability and accuracy of the predictions are discussed by comparing the states and the predicted states of experiment 1 (tracking a reference pitch rate) and 3 (reaction to a simulated elevator failure), as explained in Part I. This is done by first reiterating the purpose of the predictions, next the comparison is performed.

In order to estimate if an action is safe, the future states are predicted using the proposed action for 10 timesteps using Equation 5.1 also found in Part I. Predicting further into the future increases the uncertainty as 1) a prediction is made using a prediction as input and 2) the proposed action is applied for 10 timesteps where as in reality a different action could be applied after the initial step. Next, a check is performed whether the predicted state is inside the safe state space, $\hat{s}_{t+n} \in SSS$. If not an alternative action is proposed. Since this method is used to ensure safety, it has to be kept in mind that a false positive is undesirable, but allowed. A false negative however, is not allowed as this means the aircraft will move outside the safe flight envelope.

$$\hat{s}_{t+1} \approx s_t + \hat{F}_t \Delta s_t + \hat{G}_t \Delta a_t \tag{5.1}$$

In Figure 5.1 it can be seen how the predicted q and α values compare to the actual values near the boundary for experiment 1. It can indeed be seen that as the aircraft approaches the boundary for q the predicted values are roughly 0.2 seconds in front of the actual values. As the boundary is crossed, the safety layer proposes a different action ensuring that the actual values do not cross the boundary. As this evasive action is applied for 10 timesteps the predicted values are not representative untill the safety layer is no longer active. This can be seen in the large downward spikes of the predicted q values. Small spikes can also be seen for α. Figure 5.2 shows the predicted values of the load factor. Again it can be seen that the predicted values are approximately 0.2 seconds ahead of the actual values and again small spikes are visible. Additionally, it can be seen that the predicted values cross the actual values at (or near) the local maxima and local minima of the measured value. This shows the usability of the method as this behaviour indicates that a false negative does not occur. As the derivative of the actual value is positive in general the predicted value is above the actual value and vice versa.

For experiment 3 the same figures are displayed in Figure 5.3 and Figure 5.4. As the behaviour is more aggressive due to the failure the accuracy of the predictions as the model changes is of greater importance. It can be seen that for all states the predicted values are again ahead of the actual values and cross at, or near, the local maxima and local minima of the measured values. It can also be seen that although the system changes due to the simulated elevator failure the accuracy of the predictions stays similar.

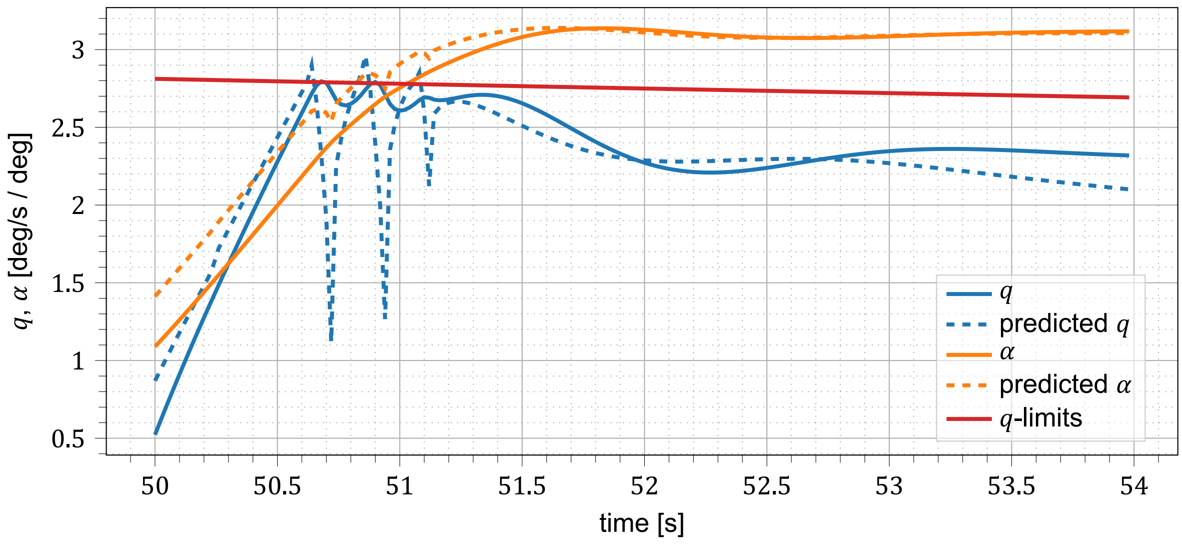

Figure 5.1: Experiment 1: predicted and actual values of q and α.

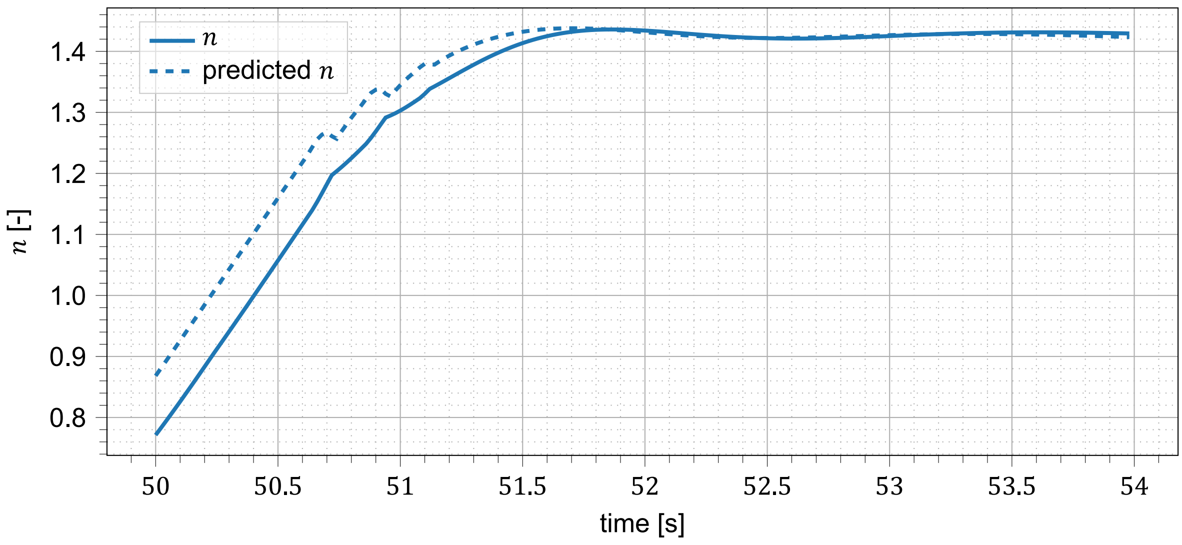

Figure 5.2: Experiment 1: predicted and actual values of n.

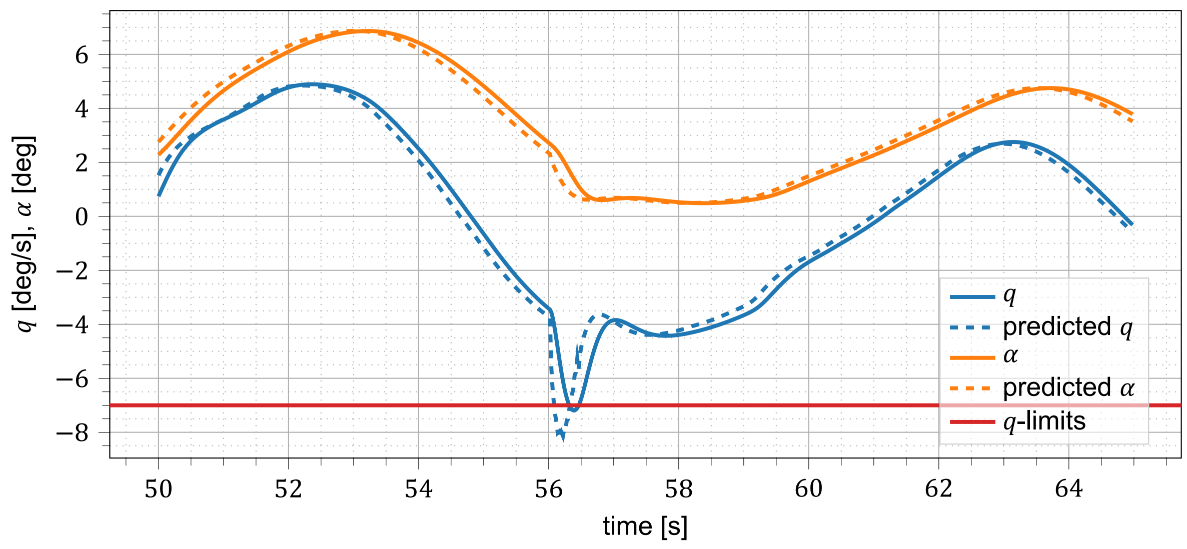

Figure 5.3: Experiment 3: predicted and actual values of q and α.

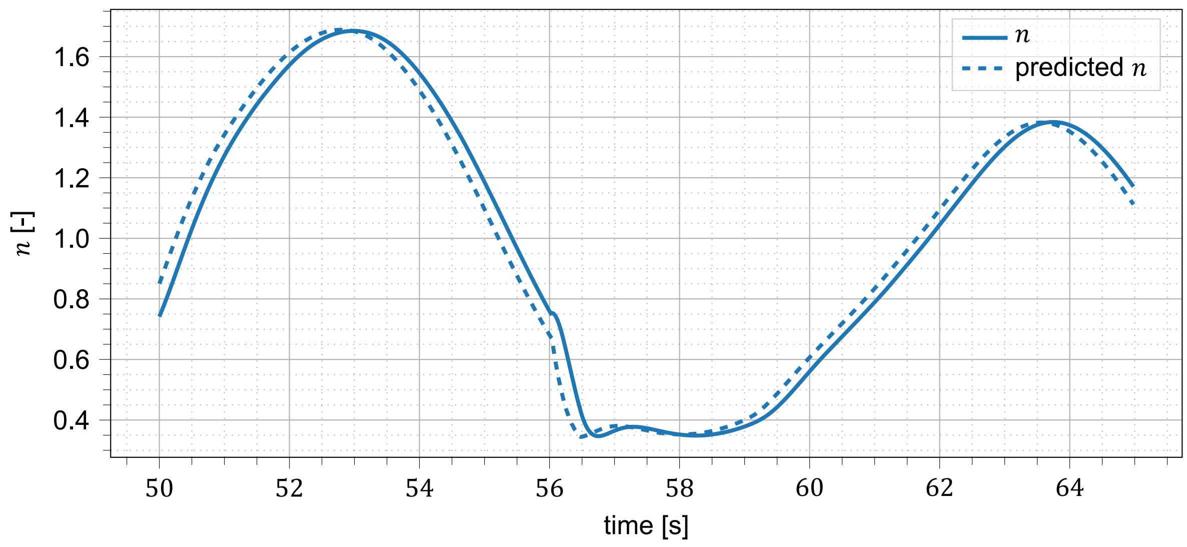

Figure 5.4: Experiment 3: predicted and actual values of n.

6 Learning Impulse

When the safety layer activates, it not only proposes a safe action, but also provides the actor network with a learning impulse. This is done by using the safety layer as a ”teacher” and emulating the proposed behaviour [56]. Via this method the controller can learn about the limit and, perhaps on the long-term, avoid it. On the short-term it avoids rapid oscillation around the boundary as control passes between the safety layer and the controller. Here the result of the learning impulse on the actor network is discussed using experiment 1 and experiment 3. As a baseline the IDHP actor network is used.

Figure 6.1 and Figure 6.2 show the weights and the biases changes for experiment 1 over time respectively. It should be noted that multiple working combinations of weights and biases exist and due to random initialization the actual value of a specific weight and bias is of less interest. The behaviour due to the learning impulse is of greater interest. In general it can be seen that, after an initial exploration process, the weight converge to a specific value. As the reference trajectory oscillates the biases also oscillates at the same frequency (0.1 Hz) in order to get the desired output.

The consequence of the learning impulse can also be seen. For the weights it is visible how IDHP looks noisy. SIDHP on the other hand displays a ”stair” pattern where the learning impulse creates the steps. For the biases similar characteristics are visible. IDHP displays what looks like a noisy square wave. SIDHP again looks less noisy, a clearer square wave is visible were the steps are provided by the learning impulse.

In Figure 6.3 and Figure 6.4 the actor weights and biases change over time for experiment 3 are visible. A similar behaviour to experiment 1 is seen, the weights seem to converge to a specific value, whereas the bias is used to create an oscillatory behaviour just as the reference. After the simulated failure the weights converge to a different specific value and the bias displays oscillatory behaviour, but at a completely different offset. The safety layer is only activated as the aircraft undergoes a simulated failure. It can be seen how the learning impulse provided, moves the weights and biases much more than in experiment 1 as the safety layer is active for much longer. As can be seen in Part I the trajectory stays smooth after control is handed back even though the actor receives a large discontinuity in state and reference. The weights and biases of IDHP can be seen to follow the same direction but slower as no impulse is provided. From the new positions IDHP and SIDHP converge to a new control policy taking into account the elevator failure.

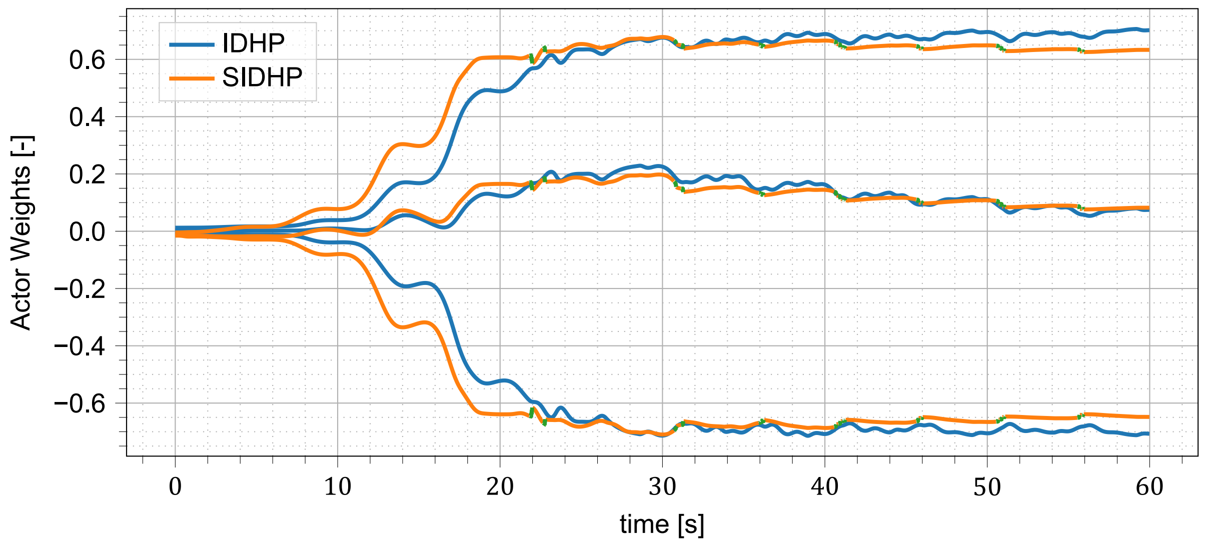

Figure 6.1: Experiment 1: change of actor weights over time. Green indicates learning due to the safety layer.

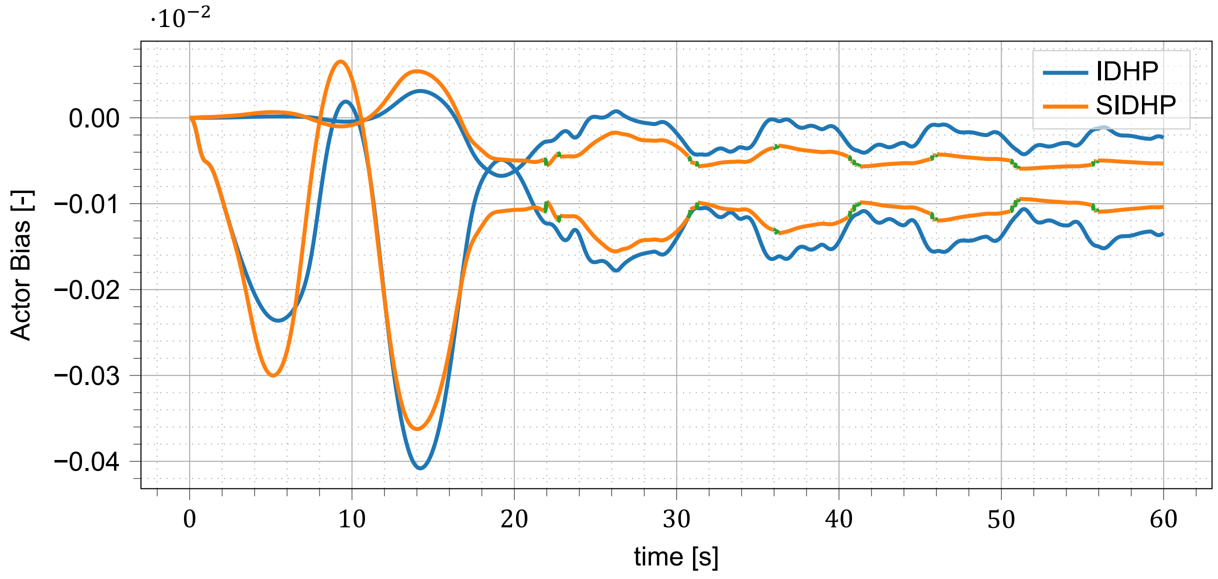

Figure 6.2: Experiment 1: change of actor biases over time. Green indicates learning due to the safety layer.

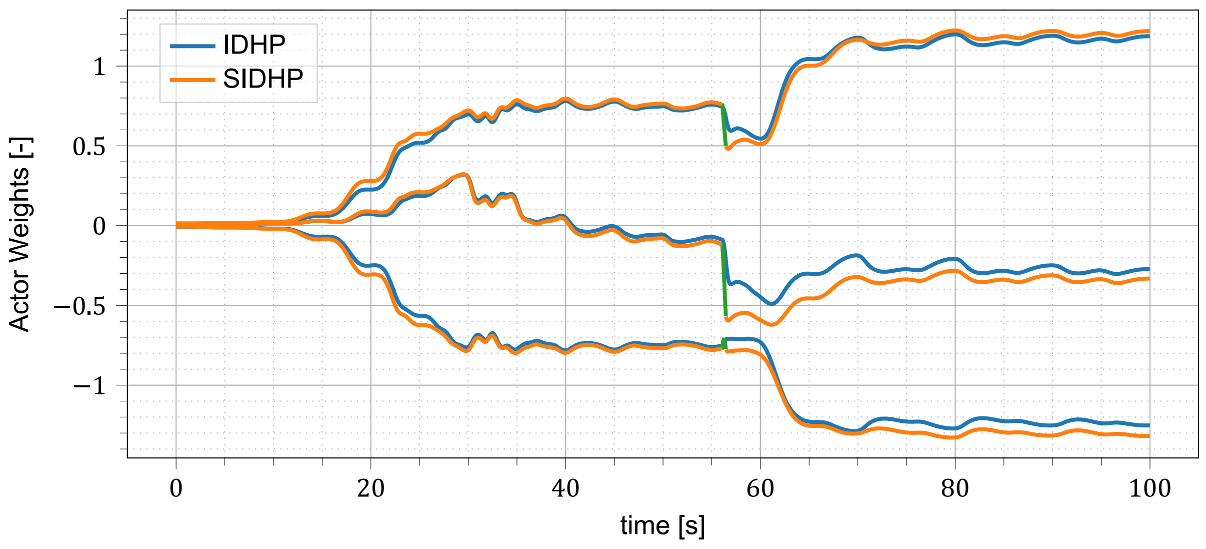

Figure 6.3: Experiment 3: change of actor weights over time. Green indicates learning due to the safety layer.

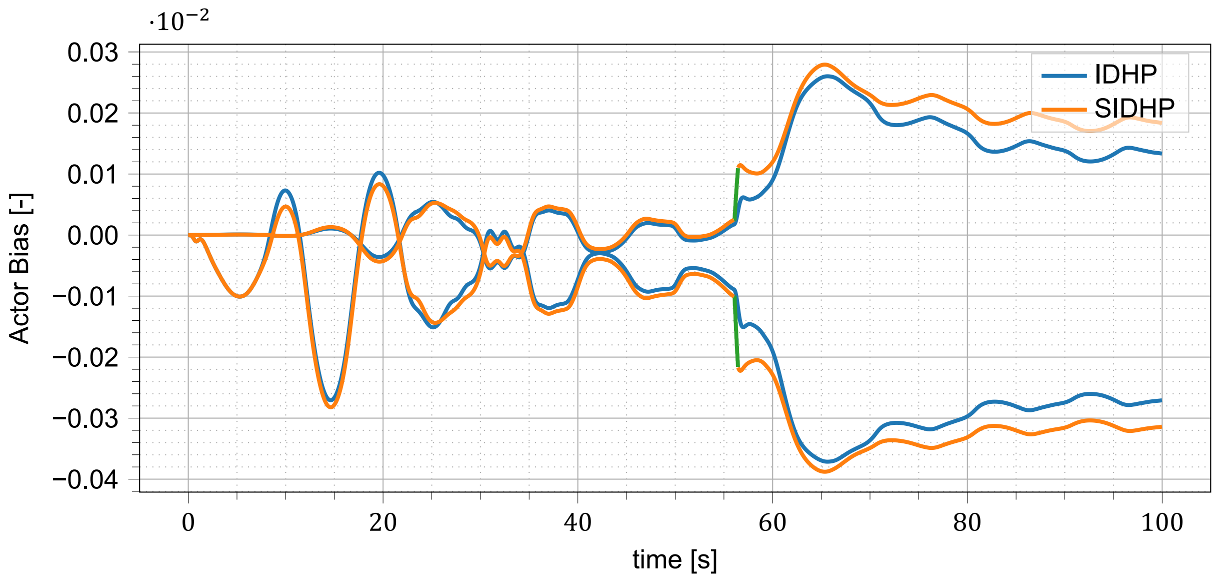

Figure 6.4: Experiment 3: change of actor biases over time. Green indicates learning due to the safety layer.

IV

Concluding Remarks

7 Conclusion

Safety in aviation has always been the highest priority. With the coming of Flight Control Systems this safety has only been enhanced. The advancements of autonomous flights and self-learning flight controllers brings exciting new possibilities, but safety guarantees are still lacking. This research aims to *increase the safety of aerospace systems through online reinforcement learning by developing a state-of-the-art safe reinforcement learning method and by comparing its performance to existing online reinforcement learning methods.* For the main research Incremental Dual Heuristic Programming (IDHP) is compared to Safe Incremental Dual Heuristic Programming (SIDHP), which consist of IDHP together with a safety layer, using a non-linear high fidelity simulation model of the Cessna Citation II.

The research objective is split into research questions. Using the research questions stated in chapter 1 the contributions of this research are discussed.

1 What is the state-of-the-art in the field of online reinforcement learning?

- What training methods are used for online reinforcement learning
- What is the state-of-the-art in the field of safe online reinforcement learning?
- What is the state-of-the-art in the field of safe online reinforcement learning in flight control?

In chapter 2, the literature overview, it is seen that an optimal policy is obtained through, in theory, infinite experience. This is required to explore all states and gain knowledge of the environment by means of the reward signal. With continuous spaces this is problematic as infinite experience is unattainable and therefore the use of function approximators is introduced. Function approximators are capable of generalizing the experience such that the information is efficiently stored and also meaningful estimates of unseen states can be made. When studying the state-of-the-art in the field of reinforcement learning Artificial Neural Networks (ANN) are identified as a powerful function approximator. From literature it is identified that in order to train online TD-learning is the most suitable training method.

Furthermore the exploration-exploitation dilemma is discussed. During exploration the agent learns through experience in order to find the optimal policy. This also makes exploration dangerous as unsafe state can be visited. Often this risk mitigated by first introducing an offline training phase such that the agent already has some knowledge of the system beforehand. Adding additional knowledge of the system to the agent using different methods can thus also increase safety as the agent does not have to experience this knowledge, increasing both safety and efficiency. For control tasks Incremental Dual Heuristic Programming (IDHP) shows most potential as it can also be applied in an online fashion without any prior knowledge of the system due to the added benefit of an incremental model.

As reinforcement learning builds upon experience it is not immediately designed for safety. However the need for safe exploration in various industries and sectors has lead to different methods using various ways to provide additional knowledge to agent such that safety is increased. From literature it is concluded that two types of safe reinforcement learning show most potential for application in adaptive flight control. First, constrained exploration reinforcement learning, were actions are constrained based on an internal model of the safe states. These safe states are either given beforehand or estimated using risk signals. It shows promising gains with only limited extra information required by the algorithm. Secondly, controller based reinforcement learning algorithms also show impressive results. Here reinforcement learning is used in combination with linear control theory. These methods require a safe baseline controller which may be difficult to obtain or simply not available. The choice for constrained exploration is made as an internal model is already available when using IDHP and therefore little extra knowledge is required. The incremental model is then used, not only during training, but also to predict future states. A safety layer is then added to ensure only safe actions are taken. This results in Safe Incremental Dual Heuristic Programming (SIDHP).

2 How is a trade-off performed between safety and performance in reinforcement learning?

- How is safety compared/measured?
- How is performance compared/measured?

As reinforcement learning is becoming more widely used, safe reinforcement learning is getting more attention from various industries and sector. Whereas fundamental reinforcement learning learns trough trial and error, safe reinforcement learning mostly tries to enhance a safe but conservative policy. Learning online without prior knowledge thus requires additional system knowledge to be added such that a safe policy can be guaranteed. Safety is measured in the number of failed runs, or the number of runs before a working policy is found. Performance is measured using the total return, or cumulative reward. Reward is defined depending on the task at hand. Also the TPM_2 metric is introduced which is proposed to be used to certify adaptive flight controllers. In tasks where safety is crucial a trade-off between safety and performance is irrelevant. Furthermore chapter 3 showed that using safety measures can also increase performance.

3 How is safety guaranteed when using reinforcement learning in flight control?

- How is safety defined in flight control?
- What (initial) information needs to be available to the system to ensure safety?
- What methods are available to ensure safety in initialization and exploration?

In order to apply reinforcement learning on flight control problems it is important that the safe flight envelope of the aircraft is adhered to as explained in chapter 2. This can be considered the safe state space of the reinforcement learning problem. This fits well with constrained exploration reinforcement learning as knowledge of the bounds are known in advance. Using the safe flight envelope in combination with constrained exploration thus leads to increased safety and robustness of the algorithm. In theory actions resulting in an unsafe state are replaced and the safe state space is thus always adhered to. In the preliminary analysis this method led to zero failures. During the research as presented in Part I SIDHP reduced the number of failures, but could not be entirely failure free. This can be attributed to the fact that high learning rates were used.

The stability of the system is also of importance as oscillations could lead to uncontrollable movements. An example of this behaviour is the response of IDHP and SIDHP when large learning rates are used. It is seen how an exploding gradient creates unstable response to the task at hand. However simply applying small learning rates does not solve the problem as the controller then becomes less capable of adapting to changing circumstances in the environment.

4 How does the performance of the selected safe online continuous reinforcement learning method compare to existing continuous reinforcement learning controllers, such as (i)ADP?

- How does safety compare?
- How adaptive is the safe reinforcement learning method compared to existing reinforcement learning controllers?

In Part I IDHP and SIDHP are compared in terms of safety, performance and adaptiveness using three different experiments. Of these characteristics safety is considered to be the highest priority. Safety is assessed by looking at the number of failed simulation runs, performance is evaluated by looking at the cumulative reward, and the adaptiveness is tested by inspecting the response to an elevator failure.

The safety of IDHP and SIDHP is tested by comparing the number of failed runs on a height tracking task. A virtual safe flight envelope is given to which the algorithms have to adhere. Multiple hyper parameter settings are selected to also test the robustness of the algorithm to these settings. Often problems can only be solved using reinforcement learning when specific hyper parameters are chosen. As an aircraft faces many different tasks this is undesirable and optimally the controller is robust to these hyper parameter settings. For learning rates up to 1 SIDHP shows increased safety as only 3 out 2700 simulation failed whereas IDHP failed in 2300 runs. For higher learning rates SIDHP is still safer than IDHP but also fails in 326 out of 2700 runs. IDHP fails in 865 runs. This number of failures are attributed to the high learning rate creating instability. SIDHP tries to slow down the learning process, but is not always capable in doing so. On succesful runs IDHP shows better tracking performance which can be explained by the extra exploration that occurs, however it is this freedom that also results in failed runs.

To test the adaptiveness IDHP and SIDHP are tasked with tracking a pitch rate reference. At a predetermined time an elevator failure is simulated resulting in a loss of trim whilst, elevator effectiveness is reduced to 50%. Both IDHP and SIDHP are able to recover from this failure and learn a new control policy, showing that SIDHP is still as adaptive as IDHP. Furthermore the learning impulse provided by the safety layer helps SIDHP in correcting the failure even after control is passed back from the safety layer to the controller.

The objective of the research is to *increase the safety of aerospace systems through online reinforcement learning by developing a state-of-the-art safe reinforcement learning method and compare its performance to existing online reinforcement learning methods.*

A new safe online continuous reinforcement learning controller for flight control is presented combining knowledge of the safe flight envelope with constrained exploration and IDHP. This results in SIDHP, a safe reinforcement learning controller for flight control capable of online learning without prior knowledge whilst adhering to a safe flight envelope. Additionally, it proves to be able to adapt to sudden changes in the environment, whilst adhering to a safe flight envelope. Lastly, it is more robust to hyper parameter settings compared to IDHP. The tracking performance of SIDHP is slightly worse due to the constrained exploration. The proposed framework can still not provide the much wanted safety guarantees needed for certification on a real aircraft, but provides a stepping stone for further research.

8 Future Work

This research has provided new knowledge on safe online continuous reinforcement learning in flight control by introducing SIDHP. As with every research the results lead to new questions and ideas for future research:

- During this study the scope was limited in order to investigate the proposed proof of concept. One of these assumptions is the fact that clean measurements of the state are used. For further research it would provide value insight to also add noise to the measurement. In particular the effect of noise on the accuracy of the state prediction used to activate the safety layer would be interesting.
- Within the current framework knowledge about the limits of the safe flight envelope is provided as the predicted values go outside the safe flight envelope. Another technique that could be applied is to also provide this information via risk signals such as in [48]. As the limits are approached the risk could increase. In this study only 10 timesteps (or 0.2 second) into the future are predicted. Not only could a risk signal provide knowledge about the limits, but it could also increase the time the safety layer has to react and provide safe actions.
- In this study it is shown that IDHP is very sensitive to the learning rate. SIDHP although more robust was also incapable of dealing with learning rates larger than 2. However low learning rate show considerably less tracking performance. As such adaptive learning rates as shown in [8] and applied in [76] can provide stability whilst also increasing performance.
- As a non-linear high-fidelity simulation was used it proved to complex to control the altitude H and flight path angle γ directly using an online method. Therefore a PID controller was used as an outer loop controller to provide a pitch rate reference. Applying cascaded networks might solve this problem as also shown in [76, 75, 74] and get rid of the PID controller outer loop. This would also allow for a more generalized control policy eventually leading to increased safety and performance.
- SIDHP has been shown to increase safety and robustness in aircraft control, however to simplify the problem only longitudinal control has been performed. Applying SIDHP on both longitudinal and lateral control would provide further insight into the performance. Instead of 1 output, the elevator, SIDHP should then provide 3 commands, the elevator, ailerons and rudder. This would also impact the method of choosing a corrective action as now the magnitude of incremental model should play a role. Therefore it is also advised to include the magnitude of the specific entries of the incremental model into the corrective action vector a_c instead of only the sign.
- In this study the longest simulation runs took less than 5 minutes. It would be interesting to see the effect of longer simulations on the incremental model and the critic and actor networks. In this study the network weights seemed to converge to a value whilst the bias kept oscillating. Other studies have encountered exploding networks when longer simulation times were used [38]. Furthermore the adaptability of the incremental model becomes less as time progresses. Therefore this should be investigated.

- To prove the concept of SIDHP identifying the safe flight envelope was kept outside the scope of this study. State-of-the-art online safe flight envelope identification method which also are able to recognize failures as found in [73, 66] combined with SIDHP would provide a complete solution and replacement for current flight envelope protection systems.

Bibliography

[1] Achiam, J. et al. “Constrained policy optimization”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 22–31.

[2] Adams, R. J. and Banda, S. S. “Robust flight control design using dynamic inversion and structured singular value synthesis”. In: *IEEE Transactions on Control Systems Technology* 1.2 (1993), pp. 80–92.

[3] Akametalu, A. K. et al. “Reachability-based safe learning with Gaussian processes”. In: *Proceedings of the IEEE Conference on Decision and Control* 2015-Febru.February (2014), pp. 1424–1431.

[4] Ammar, H. B., Tutunov, R., and Eaton, E. “Safe policy search for lifelong reinforcement learning with sublinear regret”. In: *International Conference on Machine Learning*. Vol. 37. 2015, pp. 2361–2369.

[5] Aswani, A. et al. “Provably safe and robust learning-based model predictive control”. In: *Automatica* 49.5 (2013), pp. 1216–1226.

[6] Balas, G. J. “Flight Control Law Design: An Industry Perspective”. In: *European Journal of Control* 9.2-3 (2003), pp. 207–226.

[7] Bansal, T. et al. *Emergent Complexity via Multi-Agent Competition*. Oct. 2017. URL: `http://arxiv.org/abs/1710.03748`.

[8] Barto, A. G. and Sutton, R. S. *Adaptation of learning rate parameters, Appendix C of Goal Seeking Components for Adaptive Intelligence: An Initial Assessment. Air Force Wright Aeronautical Laboratories*. Tech. rep. Avionics Laboratory Technical Report AFWAL-TR-81-1070, Wright-Patterson AFB~…, 1981.

[9] Barto, A. G., Sutton, R. S., and Anderson, C. W. “Neuronlike adaptive elements that can solve difficult learning control problems”. In: *IEEE transactions on systems, man, and cybernetics* 5 (1983), pp. 834–846.

[10] Bellman, R. “A Markovian decision process”. In: *Journal of Mathematics and Mechanics* (1957), pp. 679–684.

[11] Berkenkamp, F. and Krause, A. “Tutorial on Safe Reinforcement Learning”. In: (2018).

[12] Berkenkamp, F., Krause, A., and Schoellig, A. P. “Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics”. In: (Feb. 2016).

[13] Berkenkamp, F. and Schoellig, A. P. “Safe and robust learning control with Gaussian processes”. In: *2015 European Control Conference, ECC 2015*. 2015, pp. 2496–2501.

[14] Berkenkamp, F., Schoellig, A. P., and Krause, A. “Safe controller optimization for quadrotors with Gaussian processes”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. June. 2016, pp. 491–496.

[15] Berkenkamp, F. et al. “Safe model-based reinforcement learning with stability guarantees”. In: *Advances in Neural Information Processing Systems*. Vol. 47. 12. May 2017, pp. 908–918.

[16] Bhattacharyya, S. et al. *Certification Considerations for Adaptive Systems*. Tech. rep. 2015.

[17] Briere, D. and Traverse, P. “AIRBUS A320/A330/A340 electrical flight controls - A family of fault-tolerant systems”. In: *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*. 1993, pp. 616–623.

[18] Brockman, G. et al. *OpenAI Gym*. June 2016. URL: `http://arxiv.org/abs/1606.01540`.

[19] Collinson, R. “Fly-by-wire flight control”. In: *Computing & Control Engineering Journal* 10.4 (Aug. 1999), pp. 141–152.

[20] Crespo, L. et al. “Analysis of control strategies for aircraft flight upset recovery”. In: *AIAA Guidance, Navigation, and Control Conference*. 2012, p. 5026.

[21] Dalal, G. et al. “Safe Exploration in Continuous Action Spaces”. In: *arXiv preprint arXiv:1801.08757* (2018).

[22] Doyle, J., Lenz, K., and Packard, A. “Design examples using μ-synthesis: Space shuttle lateral axis FCS during reentry”. In: *Modelling, Robustness and Sensitivity Reduction in Control Systems*. Springer, 1987, pp. 127–154.

[23] Garcia, J. and Fernandez, F. “A Comprehensive Survey on Safe Reinforcement Learning”. In: *The Journal of Machine Learning Research* 16 (2015), pp. 1437–1480.

[24] Garcia, J. and Fernandez, F. “Safe exploration of state and action spaces in reinforcement learning”. In: *Journal of Artificial Intelligence Research* 45 (2012), pp. 515–564.

[25] Grondman, F. et al. “Design and Flight Testing of Incremental Nonlinear Dynamic Inversion-based Control Laws for a Passenger Aircraft”. In: *2018 AIAA Guidance, Navigation, and Control Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics, Jan. 2018.

[26] Gu, S. et al. *Continuous Deep Q-Learning with Model-based Acceleration*. Mar. 2016. URL: `http://arxiv.org/abs/1603.00748`.

[27] Gu, S. et al. “Continuous Deep Q-Learning with Model-based Acceleration”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. Vol. 48. 2016, pp. 2829–2838.

[28] Hans, A. et al. *Safe Exploration for Reinforcement Learning*. Tech. rep.

[29] Heess, N. et al. *Emergence of Locomotion Behaviours in Rich Environments*. 2017. URL: `http://arxiv.org/abs/1707.02286`.

[30] Heger, M. “Consideration of Risk in Reinforcement Learning”. In: *Machine Learning Proceedings 1994* (Jan. 1994), pp. 105–111.

[31] Henderson, P. et al. *Deep Reinforcement Learning that Matters*. 2017. URL: `http://arxiv.org/abs/1709.06560`.

[32] Heyer, S. *Reinforcement Learning for Flight Control: Learning to fly the PH-LAB*. Delft, University of Technology, 2019.

[33] ICAO. “State of Global Aviation Safety”. In: *International Civil Aviation* (2013).

[34] Jacklin, S. “Closing the certification gaps in adaptive flight control software”. In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2012, p. 6988.

[35] Jin, M. and Lavaei, J. “Control-Theoretic Analysis of Smoothness for Stability-Certified Reinforcement Learning”. In: *Proceedings of the IEEE Conference on Decision and Control*. Vol. 2018-Decem. 2019, pp. 6840–6847.

[36] Junell, J. et al. “Self-tuning Gains of a Quadrotor using a Simple Model for Policy Gradient Reinforcement Learning”. In: *AIAA Guidance, Navigation, and Control Conference*. January. Reston, Virginia: American Institute of Aeronautics and Astronautics, Jan. 2016.

[37] Junges, S. et al. “Safety-constrained reinforcement learning for MDPs”. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2016, pp. 130–146.

[38] Klambauer, G. et al. “Self-normalizing neural networks”. In: *Advances in neural information processing systems*. 2017, pp. 971–980.

[39] Klima, R. et al. *Robust temporal difference learning for critical domains*. Jan. 2019. URL: `http://arxiv.org/abs/1901.08021`.

[40] Klopf, A. H. *Brain function and adaptive systems-a heterostatic theory. Air Force Research Laboratories Technical Report*. Tech. rep. AFCRL-72-0164, 1972.

[41] Kretchmar, R. M. et al. “Robust Reinforcement Learning Control with Static and Dynamic Stability”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 11.15 (2001), pp. 1–25.

[42] Kroezen, D., Kampen, E.-J. van, and Chu, Q. *Online Reinforcement Learning for Flight Control*. Delft, University of Technology, 2019.

[43] Kulcsár, B. "LQG/LTR Controller design for an aircraft model". In: *Periodica Polytechnica Transportation Engineering* 28.1-2 (2000), pp. 131–142.

[44] Lillicrap, T. P. et al. *Continuous control with deep reinforcement learning*. Sept. 2015. URL: `http://arxiv.org/abs/1509.02971`.

[45] Linden, C. van der. *DASMAT-Delft University aircraft simulation model and analysis tool: A Matlab/Simulink environment for flight dynamics and control analysis*. Delft, University of Technology, 1996.

[46] Lowrey, K. et al. "Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control". In: (Nov. 2018), pp. 1–11.

[47] Lu, P. et al. "Aircraft fault-tolerant trajectory control using Incremental Nonlinear Dynamic Inversion". In: *Control Engineering Practice* 57 (2016), pp. 126–141.

[48] Mannucci, T. et al. "Safe Exploration Algorithms for Reinforcement Learning Controllers". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.4 (2018), pp. 1069–1081.

[49] Michels, J., Saxena, A., and Ng, A. Y. "High speed obstacle avoidance using monocular vision and reinforcement learning". In: *Proceedings of the 22nd international conference on Machine learning - ICML '05*. New York, New York, USA: ACM Press, 2005, pp. 593–600.

[50] Michie, D. and Chambers, R. A. "BOXES: An experiment in adaptive control". In: *Machine intelligence* 2.2 (1968), pp. 137–152.

[51] Minsky, M. L. *Theory of neural-analog reinforcement systems and its application to the brain model problem*. Princeton University., 1954.

[52] Mnih, V. et al. *Playing Atari with Deep Reinforcement Learning*. Dec. 2013. URL: `http://arxiv.org/abs/1312.5602`.

[53] Ng, A. Y. et al. "Autonomous helicopter flight via reinforcement learning". In: *Advances in neural information processing systems*. 2004, pp. 799–806.

[54] Prokhorov, D. V. and Wunsch, D. C. "Adaptive critic designs". In: *IEEE Transactions on Neural Networks* 8.5 (1997), pp. 997–1007.

[55] Reiner, J., Balas, G. J., and Garrard, W. L. "Robust dynamic inversion for control of highly maneuverable aircraft". In: *Journal of Guidance, control, and dynamics* 18.1 (1995), pp. 18–24.

[56] Schaal, S. "Learning from demonstration". In: *Advances in Neural Information Processing Systems* 9 (1997), pp. 1040–1046.

[57] Schulman, J. et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. June 2015. URL: `http://arxiv.org/abs/1506.02438`.

[58] Schulman, J. et al. *Proximal Policy Optimization Algorithms*. July 2017. URL: `http://arxiv.org/abs/1707.06347`.

[59] Schulman, J. et al. *Trust Region Policy Optimization*. Feb. 2015. URL: `http://arxiv.org/abs/1502.05477`.

[60] Sieberling, S., Chu, Q. P., and Mulder, J. A. "Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction". In: *Journal of Guidance, Control, and Dynamics* 33.6 (Nov. 2010), pp. 1732–1742.

[61] Silver, D. et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[62] Stepanyan, V. et al. "Stability and Performance Metrics for Adaptive Flight Control". In: *AIAA Guidance, Navigation, and Control Conference*. Reston, Virigina: American Institute of Aeronautics and Astronautics, Aug. 2009, pp. 1–19.

[63] Stoop, J. A. and Kahan, J. P. "Flying is the safest way to travel". In: *European journal of transport and infrastructure research* 5.2 (2005), p. 115.

[64] Sutton, R. S. “Learning to predict by the methods of temporal differences”. In: *Machine learning* 3.1 (1988), pp. 9–44.

[65] Sutton, R. S. and Barto, A. G. *Reinforcement Learning An Introduction*. 2nd. Cambridge, MA, USA: MIT Press, 2018.

[66] Tang, L. et al. “Methodologies for adaptive flight envelope estimation and protection”. In: *AIAA guidance, navigation, and control conference*. 2009, p. 6260.

[67] Thorndike, E. *Animal Intelligence, Darien, Ct*. 1911.

[68] Turing, A. M. *Intelligent machinery*. NPL. Mathematics Division, 1948.

[69] Walter, W. G. “A machine that learns”. In: *Scientific American* 185.2 (1951), pp. 60–64.

[70] Watkins, C. J. C. H. and Dayan, P. “Q-learning”. In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[71] Watkins, C. J. C. H. *Learning from delayed rewards*. King’s College, Cambridge, 1989.

[72] Werbos, P. J. “Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 17.1 (1987), pp. 7–20.

[73] Zhang, Y., Visser, C. C. de, and Chu, Q. P. “Online safe flight envelope prediction for damaged aircraft: a database-driven approach”. In: *AIAA Modeling and Simulation Technologies Conference*. 2016, p. 1189.

[74] Zhou, Y., Kampen, E. van, and Chu, Q. “Autonomous navigation in partially observable environments using hierarchical q-learning”. In: *Proceedings of the 2016 International Micro Air Vehicles Conference and Competition, Beijing, China*. 2016.

[75] Zhou, Y., Kampen, E. van, and Chu, Q. P. “Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming”. In: *Proceedings of the IAC 2017* (2017).

[76] Zhou, Y., Kampen, E. J. van, and Chu, Q. P. “Incremental model based online dual heuristic programming for nonlinear adaptive control”. In: *Control Engineering Practice* 73.July 2017 (2018), pp. 13–25.