CEAS-GNC-2022-076

CEAS EuroGNC 2022

"Conference on Guidance, Navigation and Control"

3-5 May 2022 @ Technische Universität Berlin, Germany

Counter Optimization-Based Testing of Flight Envelope Protections in a Fly-By-Wire Control Law Using Deep Q-Learning[1]

David Braun Research Associate, Institute of Flight System Dynamics, Technical University of Munich, 85748, Garching, Germany. david.braun@tum.de

Rasmus Steffensen Research Associate, Institute of Flight System Dynamics, Technical University of Munich, 85748, Garching, Germany. rasmus.steffensen@tum.de

Agnes Steinert Research Associate, Institute of Flight System Dynamics, Technical University of Munich, 85748, Garching, Germany. agnes.steinert@tum.de

Florian Holzapfel Professor and Head of the Institute of Flight System Dynamics, Technical University of Munich, 85748, Garching, Germany. florian.holzapfel@tum.de

ABSTRACT

This paper investigates the use of Deep Q-learning for counter optimization-based flight control laws testing. Deep Q-learning is a model-free Reinforcement Learning (RL) method that leverages repeated simulation of a black-box function to identify the optimal control policy. In the scope of flight control laws testing, we employ Deep Q-learning to perform worst-case analyses of highly complex nonlinear flight control systems. In particular, we show how this model-free RL method can be used to identify the worst-case combination of pilot stick inputs and wind disturbances with respect to a specified clearance criterion. Furthermore, we elaborate how the decision making of the RL agent can be constrained in order to prevent combinations of inputs that would result in maneuvers that are unfeasible or, with regard to real world application, operationally irrelevant. Ultimately, this novel approach to flight control law analysis and testing allows to assess the worst-case performance of complex closed-loop systems without the need for model simplifications. By identifying worst-case scenarios that might not be easily predicted by the control engineers, the proposed RL-based counter optimization framework can serve as a valuable tool in the design process of modern flight control laws. We demonstrate the capability of the RL-based worst-case analysis on a high-fidelity closed-loop simulation model of an all attitude turboprop demonstrator aircraft. The response of the load factor protection of the fly-by-wire control law resulting from worst-case inputs and wind gust disturbances is investigated.

Keywords: Worst-Case Analysis; Reinforcement Learning; Testing of Flight Control Laws

[1]The first three authors contributed equally to the content of this work.

Nomenclature

Symbols

$\mathscr{A}$	=	Action space
a	=	Action
h	=	Simulation step size
$\mathscr{L}$	=	Loss function
Q_π	=	Q-function
R	=	Cumulative reward
r	=	Immediate reward
$\mathscr{S}$	=	State space
s	=	State
t	=	Time
V_π	=	State value function
α	=	Learning rate
γ	=	Discount factor
δ	=	Control input
ε	=	Exploration rate
θ	=	Parameters of the neural network
Ψ, Θ, Φ	=	Euler angles
π	=	Control policy
τ	=	Smoothing factor

Acronyms

DQN	=	Deep Q-Network
FCS	=	Flight Control System
FCL	=	Flight Control Law
INDI	=	Incremental Nonlinear Dynamic Inversion
MDP	=	Markov Decision Process
NED	=	North-East-Down
NN	=	Neural Network
RL	=	Reinforcement Learning

Subindices

A	=	Aerodynamic quantity
B	=	Body-Fixed frame
cmd	=	Command value
E	=	Earth-Centered-Earth-Fixed frame
EAS	=	Equivalent Airspeed
f	=	Final time
K	=	Kinematic quantity
O	=	North-East-Down frame
W	=	Wind quantity
x, y, z	=	Cartesian coordinates
0	=	Initial time
$*$	=	Optimal quantity

1 Introduction

In view of the ever increasing levels of automation in flight control, thorough testing and validation of the closed-loop system performance is rendered imperative. Worst-case performance analysis of the closed-loop system constitutes an appealing strategy for identifying deficiencies in the Flight Control Laws (FCL). However, because of increasing system complexity, it is becoming more and more difficult for control engineers to adequately predict the worst-case scenario with respect to the closed-loop system. This is especially true for novel system architectures, such as eVTOL aircraft. Here, optimization-based worst-case analysis, referred to as counter optimization, can help to uncover unforeseen violation of safety and performance relevant criteria.

The benefit of using mathematical optimization for the purpose of flight control law clearance is demonstrated in [1]. The core idea is intriguingly simple. Instead of evaluating a clearance criterion for all possible combinations of uncertain parameters, the clearance criterion is only evaluated for the respective worst-case combination of parameters. Assuming a system fulfills the investigated test criterion under the worst-case scenario, testing for less critical conditions is rendered superfluous, which ultimately reduces the amount of required simulation and flight testing efforts. In [2], by using a multi-strategy adaptive global optimization algorithm, the worst-case analysis is extended to also include time-varying control inputs to the closed-loop system. However, discretization of the control inputs entails a quickly growing number of optimization variables, which in turn complicates the solution by means of global optimization algorithms because of the adverse effect of the curse of dimensionality. For this reason, [3] proposes the use of Optimal Control methods. Here, the restriction to linear system dynamics allows for an efficient optimization by means of Linear Programming methods. The use of linearization is justified when considering that classical Flight Control Systems (FCS) are mostly linear control laws, designed based on linear models of the aircraft dynamics. Hence, this line of research is continued in [4–6]. Linear Optimal Control-based methods have been applied to a linear longitudinal control law [7, 8] that is part of a modular FCS [9, 10] developed by the Institute of Flight System Dynamics at the Technical University of Munich. The FCS was successfully tested in flight on three different CS23 and an ultra light aircraft [11–13].

Recently, however, there is a strong trend towards nonlinear control methods, in particular Incremental Nonlinear Dynamic Inversion (INDI). These nonlinear control methods have been applied to many system configurations, including eVTOLs [14–16] and high performance aircraft [17]. As a consequence, testing of many practically relevant requirements necessitates an exhaustive nonlinear analysis. This is especially true for the counter optimization of flight envelope protection algorithms, which often show a highly nonlinear character [18, 19]. Here, an analysis based on the linearized closed-loop system is likely to be insufficient. Being faced with the task of performing a worst-case analysis of a complex nonlinear closed-loop system, subject to nonlinear flight dynamics (e.g., nonlinear 6 DoF kinematics, nonlinear aerodynamics or propulsion characteristics, actuator dynamics with saturations, etc.) or nonlinear flight control laws (e.g., switching or transition between different controller modes, control variable blending, control allocation, etc.), Optimal Control-based FCL testing is rendered very difficult in practice. This motivates the strive for novel counter optimization methods, capable of performing an exhaustive nonlinear worst-case analysis.

In this paper we open up a new line of research by investigating the use of model-free Reinforcement Learning (RL) for the purpose of counter optimization. Although the roots of RL methods are in games [20–24], many applications of RL-based methods have recently emerged in the field of Optimal Control and robotics [25, 26]. Attracted by their capability of dealing with complex black-box-type systems, this paper investigates the use of Deep Q-learning for the purpose of optimization-based testing of flight control laws. We demonstrate that model-free RL methods, in particular Deep Q-Networks (DQN) [20, 21], can be used to test a load factor protection algorithm of an INDI control law with respect to adverse conditions in terms of vertical wind gusts and aggressive longitudinal pilot stick commands.

By leveraging repeated simulation of the true nonlinear closed-loop system, we train a DQN agent to identify the combination of intended (i.e., pilot stick inputs) and unintended (i.e., wind disturbance) control inputs such that the attained load factor is maximized. This analysis provides the control designer with valuable insight into the conditions that might break the protection. Moreover, it provides confidence that the protection is able to avoid severe damage to the aircraft. In our approach, we emphasize on distinguishing solutions according to their operational relevance. Put differently, we constrain the solution space of possible stick time-histories such that only relevant solutions are provided. This effort is crucial in order to ensure that operationally relevant solutions are not overshadowed by mathematically more optimal, yet operationally irrelevant, solutions.

This paper is organized as follows. Firstly, in section 2, an introduction to Deep Q-learning is provided. Secondly, in section 3, we showcase how we cast the counter optimization problem as a RL problem. In section 4, the practical capability of the resulting RL-based counter optimization framework is demonstrated by means of a high-fidelity case study: the testing of the load factor protection of a fly-by-wire control law. Here, after providing a brief summary of the investigated nonlinear INDI control law, the response of the load factor protection algorithm for the worst-case combination of pilot inputs and wind disturbances is investigated by means of DQN-based counter optimization. Finally, in section 5, we summarize the core findings of this paper and provide an outline for future research.

2 Deep Q-Learning

Reinforcement Learning is a field of machine learning that studies how agents should behave in order to maximize the expected long-term reward. In RL, agents interact with an environment in order to “learn” how to behave optimally, i.e., which control policy π_* yields the maximum cumulative reward. The fundamental framing of RL is visualized in Fig. 1. The remainder of this section provides a brief introduction to (Deep) Q-learning and is oriented towards [27]. Readers interested in an in-depth discussion of the topic are referred to the rich literature.

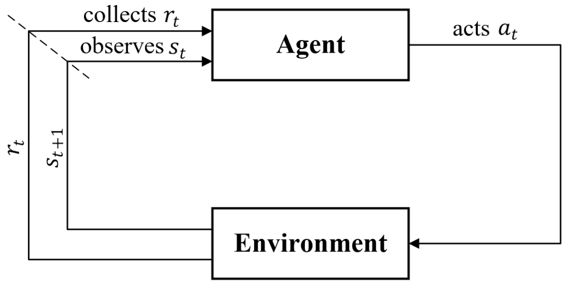

Fig. 1 Visualization of the fundamental Reinforcement Learning framework [27].

In practice, RL environments are typically modeled as Markov Decision Processes (MDP). A MDP is formally defined by a state space $\mathscr{S}$, a set of actions $\mathscr{A}$, a state transition probability $\mathbb{P}(s'|s,a)$, and a reward function $R(s,a,s')$. The transition probability $\mathbb{P}(s'|s,a)$ reflects the probability of ending up in state s' at time $t+1$ after taking action a in state s at time $t \in \mathbb{N}$. This state transition satisfies the Markov property $\mathbb{P}(s_{t+1}|s_t,a_t,s_{t-1},a_{t-1},\ldots,s_0,a_0) = \mathbb{P}(s_{t+1}|s_t,a_t)$.

At the heart of the decision making of the agent is its control policy π, which constitutes a probability distribution over actions given states:

$$\pi(a|s) = \mathbb{P}[a_t = a|s_t = s]. \tag{1}$$

Put differently, a policy is a mapping from a state $s \in \mathscr{S}$ onto the probability of taking each action $a \in \mathscr{A}$ when being in that particular state s. A policy fully and exhaustively defines the decision making of the agent. The fundamental objective of the agent is to maximize the expected discounted sum of rewards within one episode. An episode consists of a series of successive iterations t. The cumulative reward that the agent collects if he, starting in iteration t from state s_t follows a policy π, is measured by the discounted return R_t:

$$R_t = \sum_{k=0}^{T-t} \gamma^k \, r_{t+k}. \tag{2}$$

Here, r_{t+k} constitutes the immediate reward obtained in iteration $t+k$, $\gamma \in [0,1]$ is the discount rate, and T denotes the maximum number of transitions within one episode.

The value of being in state s following some policy π is reflected by the state-value function $V_\pi(s)$:

$$V_\pi(s) = \mathbb{E}[R_t | s_t = s]. \tag{3}$$

Similarly, the action-value function $Q_\pi(s,a)$ reflects the expected discounted return R_t that the agent receives if he takes action a in state s and then follows some policy π for all further steps to come. Formally, the action-value function, often referred to as Q-function, is defined as follows:

$$Q_\pi(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a]. \tag{4}$$

Outputs $Q(s,a)$ of the Q-function for a particular state-action pair (s,a) are referred to as Q-values.

By maximizing the Q-function $Q_\pi(s,a)$ for a particular state-action pair (s,a) over all possible policies π, one can determine the optimal Q-function $Q_*(s,a)$:

$$Q_*(s,a) = \max_\pi Q_\pi(s,a). \tag{5}$$

$Q_*(s,a)$ reflects the largest achievable expected discounted sum of rewards given some state-action pair (s,a). Assuming the optimal Q-function $Q_*(s,a)$ is known for all state-action pairs (s,a), the MDP is solved. In that case, the optimal policy π_* follows readily:

$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \operatorname{argmax}_{a \in \mathscr{A}} Q_*(s,a) \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Acknowledging that, $Q_*(s,a)$ allows the agent to pick the optimal action a in state s, motivates the estimation of the optimal Q-function $Q_*(s,a)$ for all possible state-action pairs (s,a).

2.1 Q-Learning

Q-learning [28] is a model-free, off-policy RL method that can be used to identify the optimal policy π_* of a MDP by computing the optimal Q-function $Q_*(s,a)$ for all state-action pairs (s,a) using value iteration. The notion "model-free" implies that Q-learning can be used without explicit knowledge of the underlying system dynamics. "Off-policy" RL algorithms learn the optimal policy π_*, referred to as "target policy", independently of the agent's action. Instead, they use a second "behavior policy" to generate experience samples (s,a,r,s') that are used to improve upon the "target policy" [27]. Often, a ε-greedy policy is used in the scope of Q-learning in order to balance exploration and exploitation. A brief summary of the Q-learning algorithm is provided in Appendix 1.

In practice, Q-learning is limited by increasingly large state and action spaces. This is particularly true for applications in the field of robotics and control where dynamic systems with continuous state or action spaces are investigated. For example, consider an application of Q-learning in the field of flight control. Here, states, such as the angle of attack or the load factor, are continuous variables that can take on an infinite number of possible values. In view of the infinite number of state-action pairs, computing and storing individual Q-values for all pairs by means of Q-learning is rendered intractable.

2.2 Deep Q-Networks

Deep Q-Network [20, 21] is a model-free Reinforcement Learning method that tackles this fundamental limitation of Q-learning by approximating the optimal Q-function using a deep Neural Network (NN). The fundamental idea of using NNs in the context of Q-learning is visualized in Fig. 2.

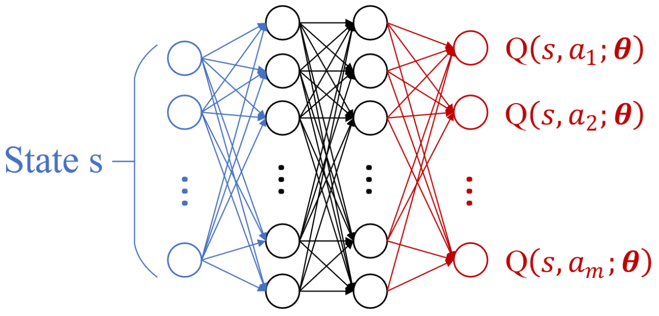

Fig. 2 Deep Neural Networks can be used as a function approximator to the Q-function. Given a state s as input, the network provides estimates $Q(s,a;\theta)$ of the Q-values for all m possible actions $a \in \mathscr{A}$.

As such, the NN acts as a function approximator to the Q-function $Q_\pi(s,a)$: Given a state s as input, it returns an estimate of the Q-value $Q(s,a;\theta)$ for all possible actions $a \in \mathscr{A}$. Essentially, the NN replaces the Q-table that is used in baseline Q-learning. As a result, DQN can cope with continuous state spaces $\mathscr{S}$. The action space $\mathscr{A}$, however, must remain discrete.

The parameters θ, i.e., the weights and biases of the NN, are parameterized by minimizing the Q-learning loss function $\mathscr{L}(\theta)$:

$$\mathscr{L}(\theta) = \mathbb{E}\Bigg[\bigg(\underbrace{r + \gamma \max_{a' \in \mathscr{A}} Q(s',a';\theta)}_{\text{approximated real return}} \quad - \quad \underbrace{Q(s,a;\theta)}_{\text{estimated Q-value}}\bigg)^2\Bigg]. \tag{7}$$

Minimization of this loss function represents the minimization of the error between the estimated Q-value and its approximated real return (i.e., its target). There are two fundamental challenges when approximating the Q-values by means of deep NNs. Firstly, the target $r + \gamma \max_{a' \in \mathscr{A}} Q(s',a';\theta)$ of the loss function depends itself on the parameters θ of the NN. Secondly, the efficiency of the learning deteriorates if highly correlated experience samples (s,a,r,s') are used for training.

Both these fundamental limitations are addressed in DQN [21]. The issue of non-stationary targets in the loss function is addressed by the use of two separate NNs with similar architecture yet different weights and biases: the "main" and the "target" network with parameters θ and $\tilde{\theta}$. The main network is used to select new actions, for example by means of a ε-greedy policy. Additionally, it is also used to compute the estimated Q-value $Q(s,a;\theta)$ in the loss function. The target network is used to compute the approximated real return $r + \gamma \max_{a' \in \mathscr{A}} Q(s',a';\tilde{\theta})$, that is the target of the loss function. Ultimately, the loss function which is used to parameterize the weights θ of the main NN in DQN states as:

$$\mathscr{L}_{\text{DQN}}(\theta) = \mathbb{E}\Bigg[\bigg(\underbrace{r + \gamma \max_{a' \in \mathscr{A}} Q(s',a';\tilde{\theta})}_{\text{computed using target NN}} \quad - \quad \underbrace{Q(s,a;\theta)}_{\text{output of main NN}}\bigg)^2\Bigg]. \tag{8}$$

Remark that only the main NN is trained by minimizing the loss function Eq. (8). The target NN is updated either periodically, by copying the parameters θ of the main NN into the target network, or continuously, by a "smooth" update, such as Polyak-like averaging:

$$\tilde{\theta} \leftarrow \tau\theta + (1-\tau)\tilde{\theta}. \tag{9}$$

Here, the target parameters $\tilde{\theta}$ are updated after each training step using a small smooth factor τ. Ultimately, the use of a dedicated target network increases the stability and efficiency of the learning process.

In order to avoid training on highly collinear data, DQN uses a technique referred to as experience replay. As such, experienced samples (s, a, r, s') are not immediately fed into a minibatch to be used for training, but only stored in a "replay buffer" that acts as a large memory. During training, the DQN loss function $\mathscr{L}_{\text{DQN}}(\theta)$ is minimized on a minibatch of experience samples by means of stochastic gradient descent. Each minibatch contains random samples from the replay buffer. By picking random samples from the replay buffer, the chance of drawing "consecutive" and therefore collinear experience samples (s, a, r, s') is significantly reduced.

In DQN, because of the maximization operation, the target $r + \gamma\ \max_{a' \in \mathscr{A}} Q(s', a'; \tilde{\theta})$ often over approximates the real return. Because Q-learning, it being a temporal difference method, learns estimates from estimates, this systematic overestimation leads to unstable and inefficient learning. In [29], building on the technique of Double Q-learning [30], a modified DQN method referred to as Double DQN is proposed. In Double DQN, the greedy selection of the action a' and the estimation of its Q-value in the target of the loss function is disentangled by using two independent NNs. While the first operation is performed by the main NN, the second operation is performed by the target NN. Recall that in vanilla DQN, the target network was used for both operations. Ultimately, Double DQN, although only requiring minor modifications to vanilla DQN, drastically improves the stability and efficiency of the Deep Q-learning algorithm.

3 Deep Q-Learning-Based Counter Optimization for Flight Control Law Testing

The fundamental objective of FCL counter optimization can be summarized as follows: Identify the sequence of time-varying inputs to the closed-loop system that minimizes the investigated testing criterion over a fixed period of time $[0, t_f]$. Note that this analysis includes both intentional (e.g., pilot stick input) and unintentional (e.g., wind disturbance) effects on the closed-loop system. In this paper, with the objective of performing a truly nonlinear worst-case analysis, we approach the counter optimization problem by means of Deep Q-learning. In particular, we employ Double Deep Q-Networks to analyze the worst-case behavior of the investigated closed-loop system.

3.1 The Reinforcement Learning Environment

Firstly, the state and action space are defined. The state space $\mathscr{S}$ must include all information that is required by the agent in order to behave optimally with respect to the objective. Recall that when using DQN, the state space can be continuous. In contrast, the action space $\mathscr{A}$ must be discrete. For the purpose of performing FCL counter optimization, the action space is defined as a finite set of input combinations. Each action $a \in \mathscr{A}$ thereby encapsulates a particular combination of closed-loop system inputs, e,g., a particular combination of pilot stick inputs and wind disturbance inputs. Moreover, discrete (failure) events such as sensor icing or actuator hardover can be considered in $\mathscr{A}$. Ultimately, the action space constitutes a discretized representation of the continuous input space of the closed-loop system. More detailed information about the choice of the state and action space is provided in subsection 4.2.

Secondly, the reward function is set up. In practice, the design of an adequate reward function constitutes one of the most challenging tasks in RL. Typically, the reward function is specifically tailored to the problem. In this paper, we use the reward for two purposes. On the one hand side, we use the reward to facilitate the learning of the optimal control policy π_*. Hence, considering the objective of performing worst-case-based analysis of the closed-loop system, we therefore reward the agent for actions that increase the likelihood of violating the investigated test criterion. On the other hand side, we use negative rewarding to teach the agent to distinguish between operationally relevant and irrelevant input sequences. In subsection 4.2, the design of the reward function, taking into account both objectives, is showcased for a practical FCL testing example.

Thirdly, the initialization and termination criterion of each episode is specified. Recalling section 2, an episode is defined as a sequence of states, actions, and rewards. For the purpose of performing FCL counter optimization, we initialize the closed-loop system to a trimmed flight condition at the beginning of each episode. After a fixed amount of time t_f has passed, the episode is terminated. Remark that the use of episodes of fixed time length $[0, t_f]$ effectively limits the time the DQN agent has to drive the system into adverse conditions. Therefore, in order to ensure that the choice of the final time does not restrict the set of relevant solutions, t_f should be chosen after consultation of domain experts.

3.2 Training of the Deep Q-Network Counter Optimization Agent

The training of the DQN agent extends over thousands of episodes. Each episode $[0, t_f]$ is discretized in time using an equidistant step size h. At the beginning of each episode, i.e., at time instance $0 \cdot h$, the state of the closed-loop system is initialized to s_0. In the scope of FCL testing, s_0 may refer to a trimmed flight condition.

In each iteration t, the agent selects an action a_t from the discrete action set $\mathscr{A}$. During training, with the objective of balancing exploration and exploitation, the decision making of the agent depends on the state observation s_t and on the exploration rate ε. Having selected an action a_t, i.e., a particular combination of inputs, the closed-loop system is simulated starting from the iteration's initial condition s_t. During the simulation of length h, the respective input values encapsulated by action a_t are applied to the closed-loop system. Next, based on the updated state observation s_{t+1}, that is the output of the closed-loop simulation, and the selected action a_t, the immediate reward r_t is computed. The obtained experience sample (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer of the DQN. Finally, the parameters θ of the main NN are updated by performing a stochastic gradient descent update on a random minibatch of experience samples, uniformly sampled from the replay buffer. Having updated the main NN, the iteration is concluded: $t \leftarrow t+1$. At this point, assuming the episode is not yet over, i.e., $t \cdot h < t_f$, the episode proceeds with the next iteration. Once an episode is concluded, the environment is reset to the initial condition s_0 and the training continues with the next episode.

A visualization of the Deep Q-learning-based counter optimization framework is presented in Fig. 3.

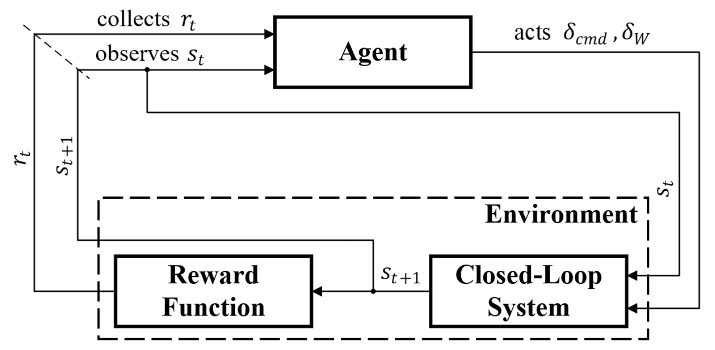

Fig. 3 Visualization of the RL-based counter optimization framework. In the scope of worst-case-based FCL testing, actions a_t constitute a particular combination of closed-loop system inputs, e.g., a particular combination of stick and wind inputs (δ_{cmd}, δ_W).

Over the course of the training the DQN agent continuously refines his control policy. Put differently, the DQN agent "learns" which actions to choose in which flight state in order to maximize its expected reward, i.e., in order to maximize the objective of the FCL counter optimization test case.

One of the main advantages of the proposed counter optimization method is the fact that the agent is trained by repeated interaction with the true, non-simplified closed-loop system. Therefore, provided that an efficient simulation interface is available, the proposed method is capable of analyzing almost arbitrarily complex closed-loop systems, including severe nonlinearities, discrete dynamics, system delays, and actuator dynamics. On the flip side, however, note that the efficiency of deep learning methods is generally very sensitive to hyperparameter tuning. Moreover, DQNs are typically best suited if the size of the action space is relatively small[1].

Ultimately, considering their capability of dealing with complex nonlinear systems, the use of model-free RL methods appears to be very promising for the purpose of counter optimization.

4 Case Study: Worst-Case Analysis of the Load Factor Protection

Finally, the capabilities of the proposed Reinforcement Learning-based counter optimization framework are demonstrated using a high-fidelity closed-loop system. In particular, with the objective of assessing the load factor protection of a nonlinear FCS, we train a DQN agent to identify the combination of pilot inputs and wind disturbances which maximizes the attained load factor.

In subsection 4.1, the investigated control laws are briefly presented. Subsequently, in subsection 4.2, we explain how the analysis of the load factor protection system can be approached from a RL perspective. Finally, in subsection 4.3, the results of the case study are presented.

4.1 The Investigated Control Laws

We consider the Incremental Nonlinear Dynamic Inversion control law presented in [17], that was developed for an all attitude turboprop demonstrator aircraft. In the following we will focus on the controller for the longitudinal motion. The control law allows blending of the control variables to provide intuitive control for the pilot over the whole flight envelope. Therefore, the pilot controls the limiting flight envelope variable at high stick deflections, i.e., angle of attack below the maneuver speed V_A and load factor above. At low stick deflections, the control variable is a blend of pitch rate and load factor similar to C^*. The block diagram of the longitudinal law is depicted in Fig. 4.

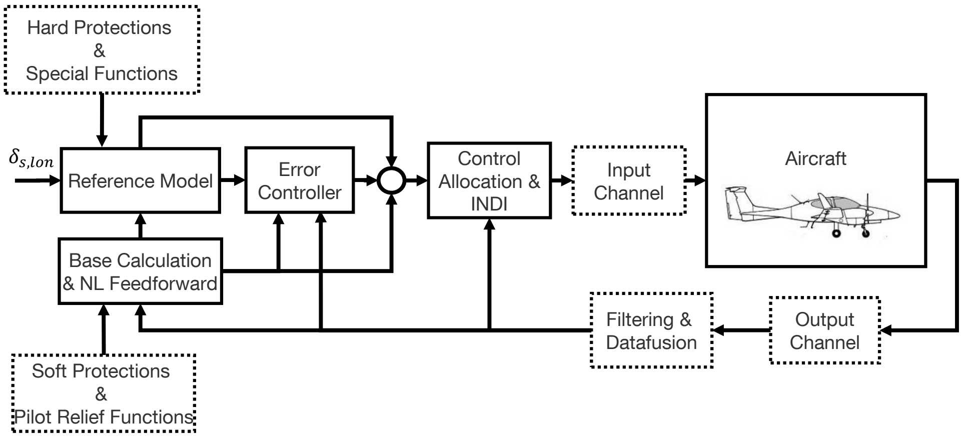
Hard Protections & Special Functions
$\delta_{s,lon}$
Reference Model
Error Controller
Control Allocation & INDI
Input Channel
Aircraft
Base Calculation & NL Feedforward
Filtering & Datafusion
Output Channel
Soft Protections & Pilot Relief Functions

Fig. 4 Schematic of the longitudinal flight control law [17].

The considered controller comprises an incremental physical reference model which shapes the pilot commands in terms of desired dynamics. An error controller ensures tracking of these commands in case of disturbances and model uncertainties. It produces a pitch acceleration command for the INDI type

[1]The authors have successfully tested the use of action spaces up to cardinality 100. The applicability of the proposed method for higher dimensional action spaces would need to be investigated in the future.

inner-loop. Finally, the commands are distributed by the control allocation to the control effectors. The filtering and datafusion contains roll-off filters, notch filters, and estimators for different entities like angle of attack. The proposed reference model structure allows to integrate hard protections, such as low energy protection, angle of attack protection, and load factor protection in a straight forward manner. Additional functions such as altitude, pitch angle and flight path angle hold, and also soft protections, such as overspeed protection, are integrated into the base value calculation, as depicted in Fig. 4. In this paper, we will focus on testing the load factor protection. For protection of the load factor two main aspects need to be considered:

1) The tracked feedback variable shall be the load factor when the load factor is close to its limit.
2) The reference load factor trajectory close to the limit shall be physically feasible with respect to aircraft and actuator dynamics.

1) The control variable blending shall ensure that the pilot controls the load factor when the aircraft is close to the load factor limit. The choice of the control variable is based on the calibrated airspeed, the current stick deflection, and a selection function with input of angle of attack and load factor. Above the manuever speed and with high stick deflections, the blend becomes a pure load factor tracking. Additionally, the selection function ensures that in all other situations, if the load factor becomes close to the maximum, the feedback variable is blended towards load factor. In that way, the load factor is always the tracked variable at the limit.
2) The stick scaling is chosen such that a full stick deflection corresponds to maximum angle of attack or load factor which ever is the most limiting. Additionally, a phase plane-based protection [18, 19] is incorporated to prevent transient overshoots in the load factor command. This is necessary since the reference model dynamics, due to handling quality requirements, have a closed-loop damping of less than 1, and hence will produce overshoots close to the limit if not limited by the phase plane-based protection.

In a classical control law, the feedback controller needs to be robust against uncertainties, perform disturbance rejection, and shape the handling qualities. These are contradicting requirements and the control law will be designed as a compromise between the three objectives. Since the above controller is separated into reference dynamics and an error controller, these can be designed with independent goals. Therefore, the design of the handling qualities (like the Gibson Dropback criterion, Control Anticipation Parameter, Transient Peak Ratio, Short Period characteristics, etc.) is decoupled from the design of reference tracking. The closed-loop reference model dynamics are tuned to satisfy specified handling qualities, while the error controller is tuned to reject disturbances and ensure tracking of the reference dynamics, also in presence of uncertainties, without deteriorating the handling qualities. The increased tracking performance also close to the limit serves as a part of the envelope protection of the aircraft. Since disturbances such as gusts enter as a feed-through on the load factor, it is possible that the actual load factor of the aircraft exceeds the limiting value. In the next section, it will be shown how DQN-based Deep Q-learning can be used to assess whether these excursions are limited to an acceptable magnitude.

For the investigation of the load factor protection, it is important to understand the philosophy of the low energy and overspeed protection. Both protections use a 6 second ahead predicted speed to evaluate if the velocity will exceed the maximum or minimum velocity. The protection is divided into an advisory and a protection region. If the aircraft is at an extreme attitude ($|\Phi| > 60^\circ$ or $\Theta < -50^\circ$ or $\Theta > 60^\circ$), only advisories are given to the pilot, and the commands remain unmodified. The respective attitude limits were determined based on pilot comments, after evaluation of the controller in the simulator. The motivation is that in extreme attitudes, the pilot is actively maneuvering the aircraft in a way that the protection would pose a nuisance to the pilot when performing agile maneuvers, since the protection is more conservative than a pilot. If the attitude is within a range of normal piloting, the protections will modify the commands to avoid excursions in the airspeed from its limit value. The protection is implemented as a soft protection, meaning that the pilot, when deemed necessary, can always override the commands of the protection.

4.2 The Utilized Reinforcement Learning Environment

Having presented the investigated FCL, the RL environment, used to test the load factor protection, is explained next. The state space $\mathscr{S}$ consists of 22 states that can be categorized into three groups. The first group comprises 16 appropriately selected closed-loop system states: the kinematic speed V_K, the Euler angles (Ψ, Θ, Φ), the body rates ω_K^{OB}, and the wind speed V_W, each consisting of three entries. Additionally, this first group includes the aerodynamic angle of attack α_A, the load factor $(n_z)_B$, the aerodynamic sideslip angle β_A, and the equivalent airspeed V_{EAS}. By observing these states, the agent is informed about the dynamic behavior of the closed-loop system. On top of that, five low- and overspeed control flags x_{flags} of the FCL are included in $\mathscr{S}$. As explained in more detail near the end of this section, this second group of states is used to penalize the agent for suggesting operationally irrelevant input sequences. Finally, the time t is included in the state space. In the following, episodes of a fixed length of $t_f = 8$ seconds are considered. The discretization step length h is chosen as 0.5 seconds. Ultimately, the state vector x, used by the agent to reason about its decision making, states as follows:

$$x = [V_K, \Psi, \Theta, \Phi, \omega_K^{OB}, V_W, \alpha_A, (n_z)_B, \beta_A, V_{EAS}, x_{flags}, t] \in \mathscr{S}. \tag{10}$$

For numeric reasons, the state vector is normalized before being fed into the DQN.

The action space $\mathscr{A}$ is spanned by two control channels: the normalized longitudinal stick input δ_{cmd} and the incremental wind command δ_W, which is used to shape the gust velocity in the z-direction of the North-East-Down (NED) frame $(V_{W,z,cmd})_O^E$.

In each iteration t, the agent can choose between 9 different actions $a_t \in \mathscr{A}$. Each action constitutes a particular combination of the possible control inputs in each channel. With regard to the longitudinal stick input δ_{cmd}, the agent can choose between three normalized stick positions:

$$\delta_{cmd} \in \mathscr{A}_{cmd} = \{-1, 0, 1\}, \tag{11}$$

i.e., pushing the stick all the way to the front, leaving it in the middle, or pulling it all the way to the back. Similarly, the agent can select between three different target gust velocities:

$$(V_{W,z,cmd})_O^E \in \{-10 \text{ m/s}, 0 \text{ m/s}, 10 \text{ m/s}\}. \tag{12}$$

In reality, the magnitude of the wind does not change instantaneously. For this reason, an incremental command scheme is implemented to control the gust velocity. Thus, instead of directly commanding $(V_{W,z,cmd})_O^E$, the agent controls the gust velocity indirectly through specification of the incremental wind command δ_W:

$$\delta_W \in \mathscr{A}_W = \{-1, 0, 1\}. \tag{13}$$

As such, the agent can choose whether the gust velocity shall increase by 10 m/s ($\delta_W = 1$), decrease by 10 m/s ($\delta_W = -1$), or whether it shall stay the same ($\delta_W = 0$). In order to account for the maximum and minimum wind speed of ± 10 m/s, commanding $\delta_W = \pm 1$ has no effect if the current gust velocity $(V_{W,z,cmd})_O^E$ is already at its respective maximum or minimum value. Irrespective of the target wind speed, the actual wind in the NED frame follows its target using a $1-\cos$ transient dynamic.

The design of the reward function must reconcile two objectives: firstly, finding a control policy which maximizes the attained load factor and secondly, preventing care-less behavior of the agent. To this end, the reward is structured into two components: an intermediate reward r_I, which is provided to the agent at the end of each non-terminal iteration, i.e., for all time instances $t \cdot h < t_f$, and a terminal reward r_T, which is provided to the agent only at the end of the episode, i.e., at time instance $t \cdot h = t_f$.

The intermediate reward is designed such that the agent is penalized for actions that result in operationally irrelevant sequences of control inputs. This is particularly important when considering that the investigated control law is designed for care-free handling, not care-less handling. As such, it is necessary to shape the behavior of the RL agent in such a way that unrealistic inputs, which would lead to maneuvers that are not feasible in real world applications, are not suggested. The philosophy of the overspeed protection is such, that during agile maneuvering, the overspeed protection can be overridden by the pilot. However, it is then the responsibility of the pilot to remain below the maximum airspeed of 280 knots. We enforce this behavior by penalizing the agent as soon as he exceeds the maximum equivalent airspeed V_{EAS}. As such, we define the intermediate reward r_I as:

$$r_I = \begin{cases} -2, & \text{if } V_{\text{EAS}} \geq 280\text{kts} \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Although it is allowed that protections activate and are overridden by the agent, it is required that the agent leaves the aircraft in a state at the end of the episode, i.e., at $t \cdot h = t_f$, where the FCL protections are able to prevent the aircraft from overspeeding with no further interaction of the agent. This is enforced by requiring that at the end of the episode, none of the protections shall be active or in advisory mode. Without this additional restriction, it would be possible for the agent to pitch the aircraft vertically down, accelerate to a speed just below the maximum airspeed, and then pull the longitudinal stick fully back. Such a maneuver, while maximizing the load factor excursion at the end of the episode, would result in a violation of the maximum airspeed right after the agent gives up control at the end of the episode. Such care-less handling is avoided by taking into account the overspeed/lowspeed predictions and protections in the terminal reward r_T. Considering the objective of testing the load factor protection, the terminal reward r_T must at the same time reward actions (δ_{cmd}, δ_W) that result in large attained load factors.

With the objective of balancing both, the objective of maximizing $(n_z)_B$ and preventing care-less behavior, we define the terminal reward r_T as:

$$r_T = \max(n_z)_B + \begin{cases} -2, & \text{if } V_{\text{EAS}} \geq 280\text{kts} \\ 0, & \text{otherwise} \end{cases} + \begin{cases} -2, & \text{if any protection flag is advisory or active} \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Note that the design of the reward function, as well as its numerical scaling, has a significant influence on the learning efficiency of the DQN.

4.3 Results of the Worst-Case Analysis of the Load Factor Protection

In the following the load factor protection is to be tested with respect to adverse conditions in terms of vertical wind gusts and aggressive pilot commands. For the following analysis, the pilot commands are restricted to the longitudinal stick commands. The objectives of the investigation are:

1) Provide insight for the control designer in terms of which conditions can break the protection.
2) Provide confidence that the protection is able to avoid severe damage to the aircraft.

The first objective of the case study is to generate worst-case scenarios that go beyond the scope of situations that can be imagined by the intuition of the control design engineers. This is especially relevant considering the advent of novel aircraft configurations, where knowledge about potential worst-case scenarios is sparse. In this case study, the designers of the investigated FCL have been asked to provide a best guess for the worst imaginable situation. The most adverse condition was imagined to be a dive in order to increase the airspeed, followed by a sharp pull of the stick to the maximum backwards position. At the time when the load factor hits the protected limit of 7.0, a vertical gust from below is initialized in order to push the load factor over the limit. The maximum achieved load factor under this input scenario is 9.3. In Appendix 2 Fig. 8, the result of the closed-loop system to this reference scenario is presented in more detail.

In the following the assumed worst-case scenario is compared to the control strategy that has been identified by the RL agent. Figure 5 provides a detailed summary of the DQN-based worst-case analysis, which was performed using the Reinforcement Learning toolbox of *MATLAB*[2].

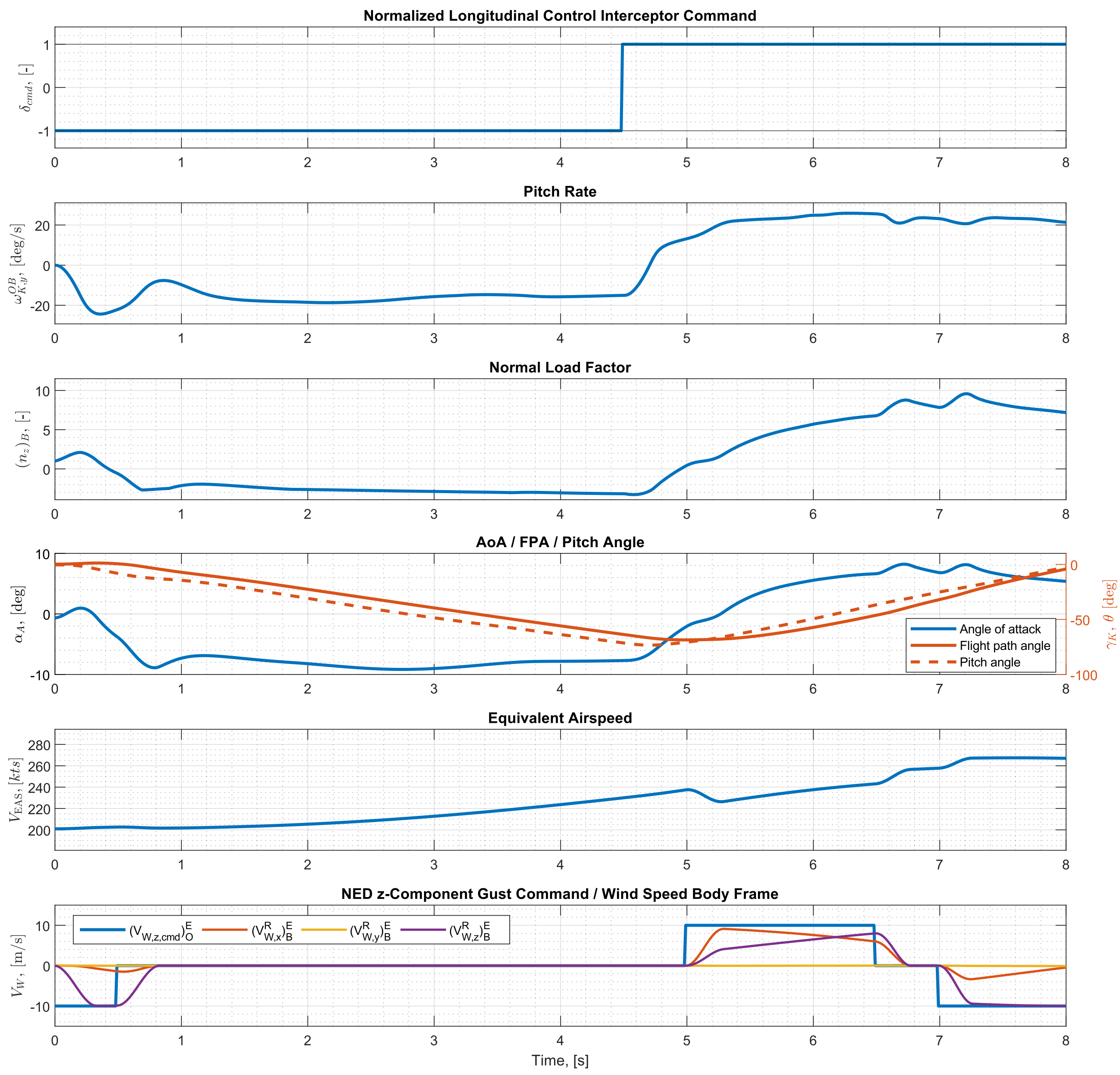

Fig. 5 Response of the closed-loop system to the worst-case combination of inputs, as identified by the DQN-based counter optimization analysis. The maximum attained load factor $(n_z)_B$ is 9.6.

When considering the identified scenario, several important observations can be made. First of all, by making a dive-pull maneuver, the agent applies a similar overall strategy. This indicates that the agent learned a reasonable control policy from a flight-mechanics perspective.

Just after initiating the pull up maneuver, the agent applies a vertical gust from above. This contributes to lower the load factor and airspeed. This is initially counter intuitive, however, it is believed that the agent is doing this to trick the protection, since the internal integrators of the control law now build up to counter act the disturbance. When the maximum load factor is reached, the agent reverses

[2]In particular the function *rlDQNAgent* was employed to train a Double DQN-based counter optimization agent.

the wind to fully exploit the feedthrough of the gust, thereby maximizing the attained load factor. The final load factor that is achieved by the maneuver and gust combination proposed by the DQN agent is 9.6. As such, the produced load factor is larger than the one resulting from the scenario provided by the control engineers.

In Fig. 6 the time histories of the FCL protection flags are plotted. It is found that the agent fully exploits the fact that the overspeed protection can be overridden, as long as at the end of the episode, i.e., after 8 seconds, neither the protection nor the advisory are active. Moreover, the agent never exceeds the maximum airspeed. Ultimately, this demonstrates that the agent is capable of distinguishing between care-free and care-less handling.

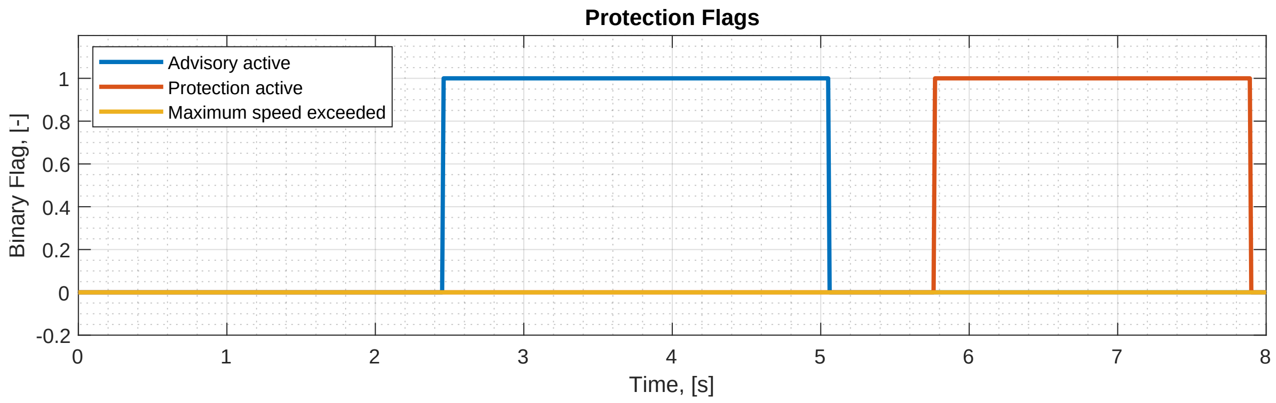

Fig. 6 Time history of the relevant FCL protection flags. These flags are used in the design of the reward function in order to penalize the agent for care-less behavior. A value of 1 signals that a flag is active.

In Fig. 7, the learning curve of the DQN agent is plotted. Here, the number of episodes is plotted on the x-axis. On the y-axis, the episodic reward and the 5-episode moving average of the reward is plotted in blue and orange color respectively. The approximated optimal Q-function value $Q_*(s_0, a)$ at the beginning of the episode is plotted in yellow color. In this case study, training is terminated after 3250 episodes. The best (i.e., the worst-case) solution is found after 1587 episodes. The authors found that the efficiency of the training and the quality of the obtained solution crucially depends on the hyperparameter tuning of the DQN agent. Among the list of relevant hyperparameters are the NN architecture, the learning rate of the stochastic gradient descent-based optimization, the number of actions in the action space $\mathscr{A}$, and the design of the reward function.

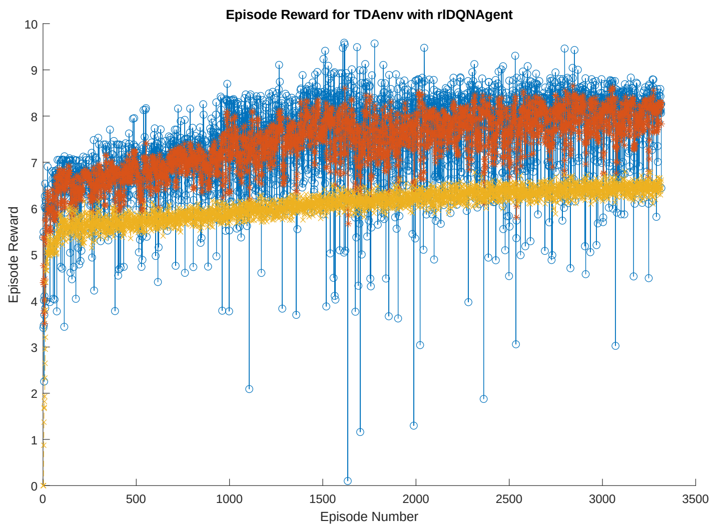

Fig. 7 Learning curve of the DQN agent: Plot of the reward per episode.

Having demonstrated that the RL-based counter optimization framework can be used to identify worst-case scenarios for complex high-fidelity systems, the second objective of the case study is addressed. As such, it can be evaluated whether the maximum attained load factor remains below the ultimate load factor of 10.5. From the results it is seen that the protection appears to be able to keep the load factor below this value, even in the case of particularly detrimental combinations of inputs. It shall be noted, however, that the RL solution is not guaranteed to be the actual worst-case situation. In order to build up confidence in the results of the RL agent, further investigation should be performed. In particular, we suggest that, for multiple test criteria, actual worst-case pilot maneuvering should be compared to the solutions provided by the RL agent. By doing this, one could empirically show that the RL agent is at least worse than any real pilot maneuvering. This experiment appears to be particularly interesting in the context of novel eVTOL configurations, where pilots have limited insight into possible worst-case scenarios.

Recall that during training, the closed-loop system is reset to the same initial condition s_0 at the beginning of each episode. As a consequence, the agent learns an optimal control policy with regard to the specified criterion for this particular initial condition of the aircraft. However, considering that the DQN constitutes a function approximator to the optimal Q-function, i.e., a mapping from a state observation onto the optimal choice of action, a trained agent can in principle transfer its decision making to different initial conditions, assuming that the same reward function still applies.

As such, we suggest that separate agents could be trained for different types of criteria, for instance for the maximization of the angle of attack, the load factor, or the angle of sideslip. Once sufficiently trained, these agents could then be used to very efficiently produce good initial guesses for worst-case scenarios. In a second step, these initial solutions could then be refined using conventional optimization methods. Additionally, with the objective of identifying the combined worst-case scenario of time-varying inputs and uncertain parameters, a two-level approach could be employed. As such, by using a pre-trained RL agent in the bottom level, uncertain parameters could be varied in the top level.

Ultimately, this case study demonstrates that RL-based counter optimization methods can be used to test modern flight control laws. In particular, the example shows that the approach is not limited to simplified, academic systems, but applicable to highly complex and nonlinear closed-loop systems.

5 Conclusion

In this paper it was investigated how model-free Reinforcement Learning techniques can be used to identify situations where the flight envelope protections are violated under the worst-case combination of pilot inputs and atmospheric disturbances. In particular, it was shown how Deep-Q Network-based Q-learning can be used to predict situations where the load factor protection within a high-fidelity simulation of a realistic flight control system is violated with longitudinal stick commands and vertical gusts. In addition, it was shown how this optimization can give confidence in the performance of the protection, with regard to preventing catastrophic damage to the aircraft. It was demonstrated that the Reinforcement Learning agents can generate scenarios that are not predictable by the control designer, and hence can serve as a valuable tool in the design process of the control laws. Further research will be conducted to expand the framework for all aspects of the control laws, including failure modes, aircraft damage, and so on. Moreover, the use of other Reinforcement Learning methods will be investigated. Finally, by comparing the obtained scenarios to actual worst-case pilot maneuvering, research will be conducted in the direction of "proving the optimality" of the solutions provided by the Reinforcement Learning agents.

Appendix

The Q-Learning Algorithm

At its core, Q-learning [28] leverages the fact that $Q_*(s,a)$ must fulfill the Bellman optimality equation:

$$Q_*(s,a) = \mathbb{E}[r + \gamma \max_{a' \in \mathscr{A}} Q_*(s',a')]. \tag{16}$$

This fundamental equation states that for any state-action pair (s,a), the expected return when starting in state s, taking action a, and then following the optimal policy π_*, must equal the sum of the immediate expected reward r, of taking action a in state s, and the discounted maximum expected reward achievable from any successor state-action pair (s',a').

At the beginning of the Q-learning algorithm, the Q-value of every state-action pair (s,a) is initialized and stored in a table. Thereafter, the algorithm iteratively improves the Q-value estimates until all Q-values have converged to the optimal Q-values. In each iteration, given the experience sample (s,a,r,s') for a particular state-action pair (s,a), the corresponding Q-function value $Q(s,a)$ is updated using the Bellman optimality equation Eq. (16). This iterative update is referred to as value iteration and formally defined as:

$$\underbrace{Q(s,a)}_{\text{new value}} \leftarrow \underbrace{Q(s,a)}_{\text{old value}} + \alpha \underbrace{\Big(\overbrace{r + \gamma \max_{a' \in \mathscr{A}} Q(s',a')}^{\text{estimated maximum reward}} - Q(s,a) \Big)}_{\text{temporal difference error}}. \tag{17}$$

The learning rate $\alpha \in (0,1]$ can be tuned in order to control how quickly the method adopts to new Q-values. Eventually, by iteration of above update law, the Q-values of each state-action pair will converge to their optimal value [27, 28].

Note that the temporal difference error, which is used to update the Q-value in Eq. (17), uses Q-function values to estimate the expected maximum reward. As such, the method is able to learn from incomplete episodes by learning estimates using estimates. In the scope of RL this technique is referred to as bootstrapping [27]. Moreover, note that the greedy action a', that is the action with the largest associated Q-function value, is used in the Q-learning value iteration. However, this does not necessarily mean that the agent will actually select the greedy action a' when interacting with the environment to generate experience samples (s,a,r,s') during training. Instead, in order to balance exploitation and exploration, a ε-greedy policy is often used as the behavior policy. As such, the agent picks the greedy action a' with probability $1-\varepsilon$ while with probability ε, a random action $a \in \mathscr{A}$ is picked. Typically, the exploration rate ε is initialized to 1 and continuously decayed throughout the remainder of the training. As a result, the initial behavior of the agent is dominated by randomness which promotes exploration. Later in the training, the agent becomes more and more greedy and will exploit the environment.

Response of the Closed-Loop System to the Assumed Worst-Case Scenario

This appendix presents the response of the closed-loop system to the assumed worst-case combination of pilot and gust inputs with respect to the load factor protection.

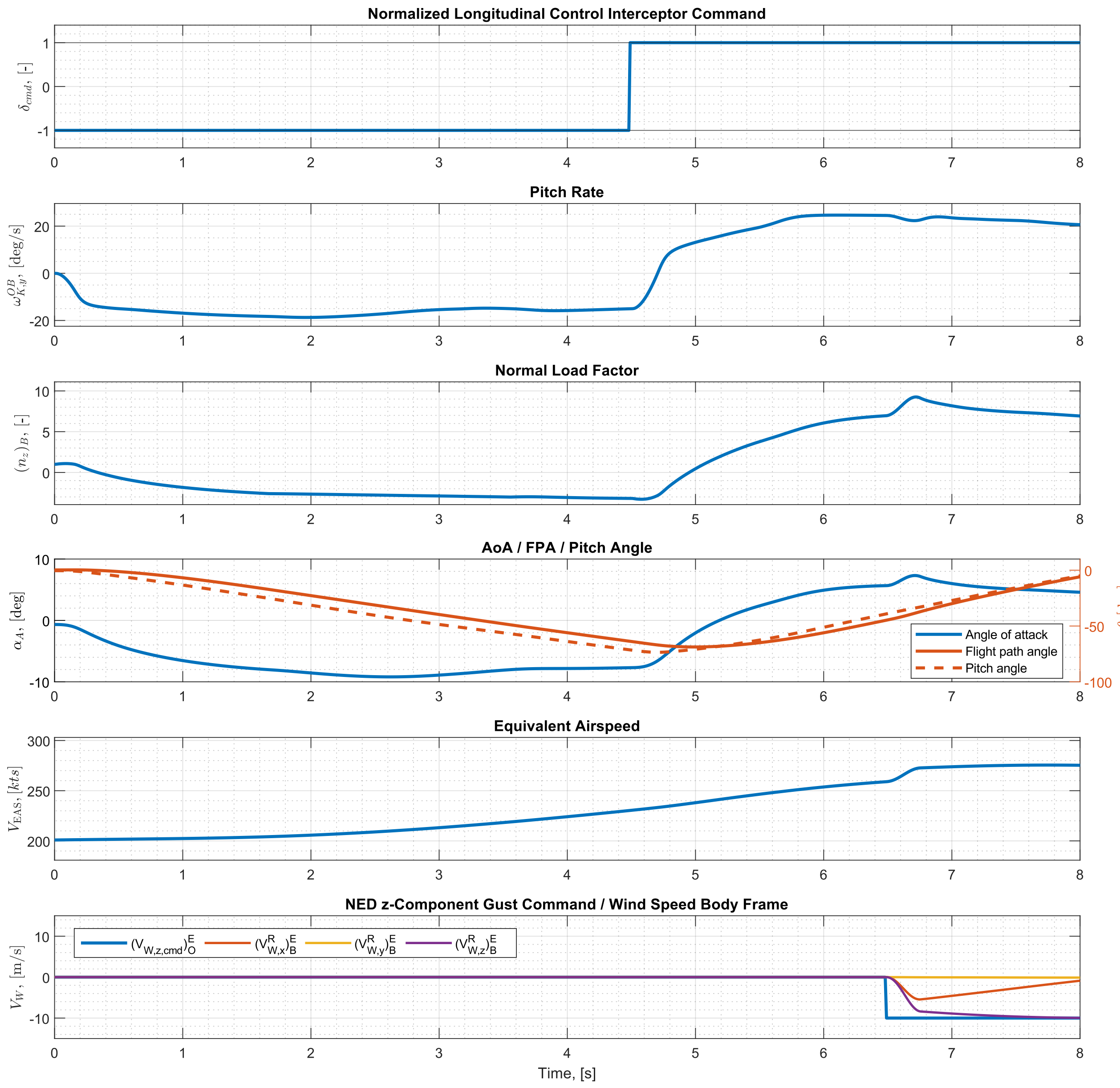

Fig. 8 Response of the closed-loop system to the assumed worst-case scenario. The maximum attained load factor $(n_z)_B$ is 9.3.

References

[1] Varga Andras, Hansson Anders, and Puyou Guilhem. *Optimization Based Clearance of Flight Control Laws*. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-22627-4.

[2] Rodney Rodríguez Robles, Miguel Boullosa, and Francisco Nieto. Flight control laws carefree handling clearance of a highly manoeuvrable aircraft using multi-strategy adaptive global optimization. 07 2017. DOI: 10.13009/EUCASS2017-204.

[3] Avriel Herrmann and Joseph Ben Asher. Flight control law clearance using optimal control theory. *Journal of Aircraft*, 1:1–15, 10 2015. DOI: 10.2514/1.C033517.

[4] Johannes Diepolder, Joseph Z. Ben-Asher, and Florian Holzapfel. Flight control law clearance using worst-case inputs under parameter uncertainty. *Journal of Guidance, Control, and Dynamics*, 43(10):1967–1974, 2020. DOI: 10.2514/1.G005236.

[5] Johannes Diepolder. *Optimal Control Based Clearance of Flight Control Laws*. Dissertation, Technische Universität München, München, 2021.

[6] Johannes Diepolder, Joseph Z Ben-Asher, Agnes Christine Gabrys, Simon P Schatz, Matthias Bittner, Matthias Rieck, Benedikt Grüter, and Florian Holzapfel. Flight control law clearance using worst-case inputs. In *ICAS 30th international congress of the international council of the aeronautical sciences*, 2016.

[7] Agnes Gabrys, Florian Holzapfel, Rasmus Steffensen, and Christian Merkl. *Flight Test Based Gain Tuning using non-parametric Frequency Domain Methods*. DOI: 10.2514/6.2021-1424.

[8] Simon P. Schatz, Agnes C. Gabrys, Daniel M. Gierszewski, and Florian Holzapfel. Inner loop command interface in a modular flight control architecture for trajectory flights of general aviation aircraft. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 86–91, 2018. DOI: 10.1109/CoDIT.2018.8394801.

[9] Erik Karlsson, Agnes Gabrys, Simon P. Schatz, and Florian Holzapfel. Dynamic flight path control coupling for energy and maneuvering integrity. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6, 2016. DOI: 10.1109/ICARCV.2016.7838678.

[10] Erik Karlsson, Simon P. Schatz, Thaddäus Baier, Christoph Dörhöfer, Agnes Gabrys, Markus Hochstrasser, Christoph Krause, Patrick J. Lauffs, Nils C. Mumm, Kajetan Nürnberger, Lars Peter, Volker Schneider, Philip Spiegel, Lukas Steinert, Alexander W. Zollitsch, and Florian Holzapfel. Automatic flight path control of an experimental da42 general aviation aircraft. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6, 2016. DOI: 10.1109/ICARCV.2016.7838566.

[11] Simon P. Schatz, Volker Schneider, Erik Karlsson, Florian Holzapfel, Thaddäus Baier, Christoph Dörhöfer, Markus Hochstrasser, Agnes Gabrys, Christoph Krause, Patrick J. Lauffs, Nils C. Mumm, Kajetan Nürnberger, Lars Peter, Philip Spiegel, Lukas Steinert, and Alexander W. Zollitsch. Flightplan flight tests of an experimental da42 general aviation aircraft. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6, 2016. DOI: 10.1109/ICARCV.2016.7838646.

[12] Agnes Steinert, Rasmus Steffensen, Daniel Gierszewski, Moritz Speckmaier, Florian Holzapfel, Robert Schmoldt, Frank Demmler, Ulrich Schell, Markus Ornigg, and Marius Koop. Experimental results of flight test based gain tuning. In *AIAA SCITECH 2022 Forum*. DOI: 10.2514/6.2022-2296.

[13] Simon P. Scherer, Moritz Speckmaier, Daniel Gierszewski, Agnes Steinert, and Florian Holzapfel. Compensation of nonlinear transmission effects in electro-mechanical flight control systems. In *AIAA SCITECH 2022 Forum*. DOI: 10.2514/6.2022-0621.

[14] Venkata Sravan Akkinapalli and Florian Holzapfel. Incremental dynamic inversion based velocity tracking controller for a multicopter system. In *2018 AIAA Guidance, Navigation, and Control Conference*. DOI: 10.2514/6.2018-1345.

[15] Ewoud JJ Smeur, Guido CHE de Croon, and Qiping Chu. Cascaded incremental nonlinear dynamic inversion for mav disturbance rejection. *Control Engineering Practice*, 73:79–90, 2018. DOI: https://doi.org/10.1016/j.conengprac.2018.01.003.

[16] Rafael A Cordeiro, R Azinheira, and Alexandra Moutinho. Cascaded incremental backstepping controller for the attitude tracking of fixed-wing aircraft. In *5th CEAS Conference on Guidance, Navigation and Control*, 2019.

[17] Rasmus Steffensen, Agnes Steinert, and Florian Holzapfel. Longitudinal incremental reference model for fly-by-wire control law using incremental non-linear dynamic inversion. In *AIAA SCITECH 2022 Forum*. DOI: 10.2514/6.2022-1230.

[18] Rasmus Steffensen, Agnes Gabrys, and Florian Holzapfel. Flight envelope protections using phase plane limits and backstepping control. In *5th CEAS Conf. Guid. Navig. Control*, 2019.

[19] Agnes Christine Gabrys, Rasmus Steffensen, Rafael de Angelis Cordeiro, José Raul Azinheira, Alexandra Moutinho, and Florian Holzapfel. Integration of phase plane flight envelope protections in cascaded incremental flight control. *IFAC-PapersOnLine*, 52(12):429–435, 2019. 21st IFAC Symposium on Automatic Control in Aerospace ACA 2019. DOI: https://doi.org/10.1016/j.ifacol.2019.11.281.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. DOI: https://doi.org/10.48550/arXiv.1312.5602.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. DOI: https://doi.org/10.1038/nature14236.

[22] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 10 2017. DOI: 10.1038/nature24270.

[23] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019. DOI: https://doi.org/10.1038/s41586-019-1724-z.

[24] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019. DOI: https://doi.org/10.48550/arXiv.1912.06680.

[25] Jens Kober, J. Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 09 2013. DOI: 10.1177/0278364913495721.

[26] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 561–591. PMLR, 29–31 Oct 2018. DOI: https://doi.org/10.48550/arXiv.1809.07731.

[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[28] Christopher Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 05 1992. DOI: 10.1007/BF00992698.

[29] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016.

[30] Hado van Hasselt. Double q-learning. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 2613–2621, Red Hook, NY, USA, 2010. Curran Associates Inc.