Soft Actor-Critic Deep Reinforcement Learning for Fault-Tolerant Flight Control

Killian Dally[*], Erik-Jan van Kampen[†]

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

Fault-tolerant flight control faces challenges, as developing a model-based controller for each unexpected failure is unrealistic, and online learning methods can handle limited system complexity due to their low sample efficiency. In this research, a model-free coupled-dynamics flight controller for a jet aircraft able to withstand multiple failure types is proposed. An offline-trained cascaded Soft Actor-Critic Deep Reinforcement Learning controller is successful on highly coupled maneuvers, including a coordinated 40°-bank climbing turn with a normalized Mean Absolute Error of 2.64%. The controller is robust to six failure cases, including the rudder jammed at −15°, the aileron effectiveness reduced by 70%, a structural failure, icing and a backward c.g. shift as the response is stable and the climbing turn is completed successfully. Robustness to biased sensor noise, atmospheric disturbances, and to varying initial flight conditions and reference signal shapes is also demonstrated.

Nomenclature

$\mathbf{s}, \mathbf{a}$	=	environment state and actor action vectors
n, m	=	number of environment states and actor actions, respectively
$\tilde{r}(\mathbf{s}, \mathbf{a})$	=	instantaneous reward function
$t, \Delta t, N$	=	discrete time-step subscript, sample time and number of samples
$f(\mathbf{s}, \mathbf{a})$	=	state transition function
$\pi, \pi^*, \pi_{\boldsymbol{\theta}}$	=	policy, optimal policy and parameterized policy approximation
$Q^\pi, Q_{\mathbf{k}}$	=	action-state value function (Q-function) and parameterized Q-function approximation
$\boldsymbol{\theta}, \mathbf{k}, \bar{\mathbf{k}}$	=	policy, Q-function and target Q-function parameter vectors
γ	=	discount factor
η	=	temperature parameter
$\mathcal{H}, \tilde{\mathcal{H}}$	=	entropy and target entropy
$L(x), J(x)$	=	loss and objective functions for variable x
$\mathcal{D}, \mathcal{B}$	=	memory buffer and minibatch (subset of the memory buffer)
$\boldsymbol{\xi}$	=	noise vector
$\mathcal{N}$	=	standard normal distribution
$\boldsymbol{\mu}, \boldsymbol{\sigma}$	=	mean and Sample Standard Deviation (SSD) vectors
λ	=	learning rate
τ	=	smoothing factor
$\mathbf{x}, \mathbf{u}$	=	aircraft state and control input vectors
$p, q, r, \phi, \theta, \psi$	=	roll, pitch and yaw rates, and roll, pitch and yaw angles
V, α, β	=	total airspeed, angle-of-attack and sideslip angle
$h, \Delta h$	=	altitude and altitude tracking error
$\delta_{\mathrm{e}}, \delta_{\mathrm{a}}, \delta_{\mathrm{r}}$	=	control surface deflections (elevator, aileron, rudder)
R	=	reference signal superscript
$\Delta\mathbf{u}, \Delta\theta^R$	=	control input and reference pitch angle increments
$\mathbf{e}, \mathbf{c}$	=	error and error cost vectors
l	=	number of units per hidden layer
C_L, C_D, C_m	=	lift, drag and pitching moment coefficients
T, A	=	signal period and amplitude

[*] M.Sc., Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology

[†] Assistant Professor, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology

I. Introduction

In-flight loss of control was the cause of 61% of commercial flight accident casualties between 2009 and 2018, indicating the need for more fault-tolerant control systems [1]. At the same time, the advent of personal air vehicles in dense urban areas calls for the development of autonomous flight controllers that can withstand multiple types of failures.

Until now, flight control automation techniques have relied on gain-scheduling to switch between parallel linear controllers tuned at specific known operating points [2]. Because of their reliance on known plant dynamics, they cannot deal with sudden changes in dynamics such as failures. Model-free adaptive and intelligent control techniques offer the possibility of replacing this inconvenient controller structure with a more general and fault-tolerant approach.

Reinforcement Learning (RL) is a bio-inspired machine learning framework for intelligent control that can offer fault-tolerance. An RL agent learns through trial-and-error by interacting with the plant, also known as the environment [3]. In its original form, RL was a tabular method with discrete action and state spaces, well suited for gaming environments. To combat the curse of dimensionality encountered when making the action and state spaces larger [4], function approximators, typically Neural Networks (NNs), can be used with the actor-critic agent structure to enable continuous control.

A common approach to actor-critic RL structures is Approximate Dynamic Programming (ADP). It has been applied to coupled-dynamics flight control for a business jet aircraft [5] and a helicopter [6] in simulations, yet all based on a known model of plant dynamics and thereby limiting their applicability to a wide range of unforeseen failures. Recent research using online incremental model-learning ADP techniques (iADP) has managed to provide adaptive control while eliminating all model dependence. Longitudinal control was proposed first by [7], and then the introduction of Incremental Dual Heuristic Programming (IDHP) demonstrated fault-tolerance on simple failure cases [8]. Using IDHP for coupled-dynamics body rate control, [9] showed that that a business jet aircraft could be controlled successfully. It was extended to altitude and attitude control with pre-tuned PID controllers in [10] and [11], yet as the longitudinal and lateral motions were decoupled severe failures cases where the coupling effects become too dominant were not tested. When the outer-loop PIDs were replaced with IDHP agents for longitudinal control in [12], a failure rate of 24% suggested that this method alone does not yet have the reliability and sample efficiency to control online the outer-loop coupled dynamics of 6-DoF systems.

A novel approach to actor-critic structures has recently been introduced, known as the field of Deep Reinforcement Learning (DRL). Enabling end-to-end offline learning, high-dimensional input spaces such as images were used to surpass human performance on multiple Atari games, as shown by [13] with Deep Q-Network (DQN) for discrete action spaces. DRL was extended to control applications by [14] with the off-policy Deep Deterministic Policy Gradient (DDPG) algorithm thanks to its continuous control abilities. DDPG flight control applications have been limited to small-scale flying-wings [15] and quadcopters [16–18]. On-policy RL algorithm Proximal Policy Optimization (PPO), proposed by [19], has aimed at reducing DDPG's learning instability and showed improved policy convergence for a quadcopter UAV in [20] and for an unmanned flying-wing aircraft in [21]. State-of-the-art Twin-Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC) have focused on reducing DDPG's critic overestimation bias [22, 23]. Unlike TD3, Soft Actor-Critic (SAC) uses a stochastic policy which was shown to encourage exploration and increase sample efficiency [23]. While it is unclear from the authors of TD3 and SAC which performed best, an independent study found that SAC outperformed TD3 in terms of sample efficiency on 4 out of 5 complex control tasks [24]. On a quadcopter control task, it recovered from unfavorable initial conditions in [25], demonstrating high robustness. SAC is identified as the most promising DRL algorithm for this flight control task. Despite DRL's ability to learn highly complex tasks, it has not been tested on coupled-dynamics flight control tasks for fixed-wing aircraft. Furthermore, due to its relatively long training time, online learning is expected to be difficult. For this reason, fault-tolerance is mainly achieved through robust control. SAC's generalization ability to multiple types of failures is unknown at this point and is to be better understood.

The contribution of this research is to advance state-of-the-art fault tolerant flight control methods by developing a model-free coupled-dynamics flight controller for a jet aircraft that can withstand multiple types of unexpected failures. This research explores the use of DRL controllers for CS-25 class aircraft generally only employed for small-scale UAVs. For this research, a high-fidelity simulation model of the Cessna Citation 500 will be used, paving the way for future test flights on the PH-LAB research aircraft thanks to its experimental fly-by-wire flight control system.

The foundations of RL and the SAC algorithm are explained in Section II, followed by a motivation for the controller design in Section III. The results are discussed in Section IV and the conclusions are presented in Section V.

II. Fundamentals

This section introduces the learning framework used in this research, actor-critic RL and the RL algorithm at hand.

A. Reinforcement Learning Problem

The actor-critic RL framework is composed of an agent that applies action $\mathbf{a}_t \in \mathbb{R}^m$ on an environment with state $\mathbf{s}_t \in \mathbb{R}^n$ at discrete time step t. The next state is determined by a state-transition function unknown to the agent in Eq. (1). It is assumed to have the Markov property, which implies that the environment's history is fully explained by its present state. The agent chooses actions based on the actor's policy, a mapping from the state space $\mathbb{R}^n$ to the action space $\mathbb{R}^m$. If it is stochastic, the action is sampled as shown in Eq. (2). The environment gives a scalar reward $\tilde{r}(\mathbf{s}_t, \mathbf{a}_t) \in \mathbb{R}$ to the agent as a feedback of its action $\mathbf{a}_t$ at each time step. The goal is to learn a policy that maximizes the reward over all states.

A critic, defined by an action-state value function, or Q-function, is introduced in Eq. (3) to characterize how beneficial it is to be in a given state $\mathbf{s}_t$ in terms of future expected reward when taking action $\mathbf{a}_t$ and following the policy thereafter. A discount factor, γ, is used to trade-off immediate and future rewards. The episode comprises of N time steps.

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t) \qquad (1) \qquad\qquad \mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t) \qquad (2) \qquad\qquad Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathop{\mathbb{E}}_{\mathbf{a}_{t+i} \sim \pi} \left[\sum_{i=0}^{N} \gamma^i \tilde{r}(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}) \mid \mathbf{s}_t, \mathbf{a}_t \right] \quad (3)$$

B. Soft Actor-Critic Algorithm

Soft Actor-Critic (SAC) is a novel off-policy DRL algorithm and an extension of the DDPG algorithm that aims at optimizing a stochastic policy [23]. Unlike DDPG's deterministic policy, a stochastic policy ensures better exploration and was found to be applicable to broader types of control tasks [17]. At evaluation time, the mean of the policy distribution is selected to make actions deterministic and ensure consistent performance.

1. Entropy

SAC adds an entropy term to the standard optimal policy expression in Eq. (4), which is a measure of the randomness in its probability distribution. Policy distributions more spread over the action space have higher entropy, which can be measured with the log-likelihood according to Eq. (5). The entropy is traded-off against future rewards with the temperature parameter η[†].

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\mathbf{a}_{t+i} \sim \pi} \left[\sum_{i=0}^{N} \gamma^i \Big(\tilde{r}(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}) + \eta \mathcal{H}\big(\pi(\cdot \mid \mathbf{s}_{t+i})\big) \Big) \right] \quad \forall\, \mathbf{s}_t \in \mathbb{R}^m \tag{4}$$

$$\mathcal{H}(\pi(\cdot \mid \mathbf{s}_t)) = \mathop{\mathbb{E}}_{\mathbf{a}' \sim \pi} [-\log \pi(\mathbf{a}' \mid \mathbf{s}_t)] \tag{5}$$

In other words, this objective favors the most random policy that still achieves a high return. This creates an inherent exploration mechanism that also prevents premature convergence to local optima. Multiple control strategies that achieve a near-optimal reward are captured, allowing for more robustness to disturbances.

2. Critic

The Q-function critic is modeled as a feed-forward Deep Neural Network (DNN) with parameter vector $\mathbf{k}$. The standard Bellman equation is modified with the expression of the entropy found in Eq. (5) to obtain a recursive expression of the soft Q-function in Eq. (6).

$$Q_{\mathbf{k}}(\mathbf{s}_t, \mathbf{a}_t) = \mathop{\mathbb{E}}_{\mathbf{a}_t, \mathbf{a}_{t+1} \sim \pi_\theta} \left[\tilde{r}(\mathbf{s}_t, \mathbf{a}_t) + \gamma \left(Q_{\mathbf{k}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \eta \log \pi_\theta(\mathbf{a}_{t+1} \mid \mathbf{s}_{t+1}) \right) \right]. \tag{6}$$

Given that SAC is an off-policy algorithm, transition samples ($\mathbf{s}_t$, $\mathbf{a}_t$, r_t, $\mathbf{s}_{t+1}$) can be collected in a memory buffer $\mathcal{D}$ and reused at a later stage. This can help ensure training samples are more independent and identically distributed, an assumption of the update method of RL algorithms. By sampling a minibatch $\mathcal{B}$ from the memory buffer, a minibatch

[†] The symbol η is used as temperature parameter instead of α in [23], the original SAC paper, to avoid confusion with the angle-of-attack.

gradient update can be performed instead of the computationally inefficient and noisy stochastic gradient update of on-policy algorithms.

To increase learning stability, a target network with parameter $\bar{\mathbf{k}}$ for the Q-function is introduced to make the gradient update follow a more constant direction. From time to time, the target network is synchronized with the current value network with an exponentially weighted moving average as a soft update mechanism, such that the target network is a delayed version of the current value network regulated by smoothing factor τ. The target network is used in the mean squared Bellman error for the Q-function loss function shown in Eq. (7). In an effort to reduce DDPG's overestimation bias of the Q-function, SAC makes use of the double Q-function trick by learning two approximators and using a pessimistic bound over the two. Transition samples are contained in minibatch $\mathcal{B}$ but because SAC is off-policy, fresh actions $\mathbf{a}_{t+1}$ can be sampled from the current policy to compute the Q-function targets.

$$L_Q\left(\mathbf{k}_i, \mathcal{B}\right) = \underset{\substack{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \sim \mathcal{B} \\ \mathbf{a}_{t+1} \sim \pi_{\boldsymbol{\theta}}}}{\mathbb{E}} \left[\left(Q_{\mathbf{k}_i}(\mathbf{s}_t, \mathbf{a}_t) - \left(\tilde{r}(\mathbf{s}_t, \mathbf{a}_t) + \gamma \left(\min_{i=1,2} Q_{\bar{\mathbf{k}}_i}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \eta \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_{t+1} \mid \mathbf{s}_{t+1}) \right) \right) \right)^2 \right] \tag{7}$$

3. Policy

The policy, or actor, is modeled as an m-dimensional multivariate Gaussian distribution with a diagonal covariance matrix. Its actions are passed to a tanh squashing function to ensure they are defined on a finite bound. The mean vector $\boldsymbol{\mu_\theta}$ and the covariance matrix, or, in this case, vector $\boldsymbol{\sigma^2_\theta}$, are estimated for each state by a DNN with parameter vector $\boldsymbol{\theta}$.

Unlike DDPG's deterministic policy, no target policy is needed as the policy's stochasticity has a smoothing effect. The stochasticity also means that the policy objective in Eq. (4) depends on the expectation over actions and is therefore non-differentiable. A reparameterization trick is proposed by the SAC authors in [23] using the known mean and standard deviation of the stochastic policy along with independent noise vector $\boldsymbol{\xi}$, and applying the squashing function, as shown in Eq. (8). A policy objective with an expectation over noise instead of actions and making use of the Q-function as an approximation of expected future rewards and entropy is introduced in Eq. (9).

$$\tilde{\mathbf{a}}_{\boldsymbol{\theta}}(\mathbf{s}_t, \boldsymbol{\xi}) = \tanh\left(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{s}_t) + \boldsymbol{\sigma}_{\boldsymbol{\theta}}(\mathbf{s}_t) \odot \boldsymbol{\xi}\right), \quad \boldsymbol{\xi} \sim \mathcal{N}(\vec{\mathbf{0}}, \vec{\mathbf{1}}) \tag{8}$$

$$J_\pi\left(\boldsymbol{\theta}, \mathcal{B}\right) = \underset{\substack{\mathbf{s}_t \sim \mathcal{B} \\ \boldsymbol{\xi} \sim \mathcal{N}}}{\mathbb{E}} \left[\min_{i=1,2} Q_{\mathbf{k}_i}(\mathbf{s}_t, \tilde{\mathbf{a}}_{\boldsymbol{\theta}}(\mathbf{s}_t, \boldsymbol{\xi})) - \eta \log \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{a}}_{\boldsymbol{\theta}}(\mathbf{s}_t, \boldsymbol{\xi}) \mid \mathbf{s}_t) \right] \tag{9}$$

4. Automatic Temperature Adjustment

SAC can be unstable with respect to temperature parameter η, so the latter was proposed to be controlled automatically in [26]. Optimal entropy is not constant throughout training as the need for exploration is expected to decrease with increasing training steps. A loss function $L\left(\eta\right)$ is introduced in Eq. (10) to dynamically find the lowest temperature that still ensures a certain minimum target entropy $\bar{\mathcal{H}}$ while maximizing the return. A good empirical value for the target entropy is found to be connected to the action space dimension with $\log \bar{\mathcal{H}} = -m$ [26].

$$L\left(\eta\right) = \underset{\substack{\mathbf{s}_t \sim \mathcal{B} \\ \mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}}}{\mathbb{E}} \left[-\eta \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t) - \eta \bar{\mathcal{H}} \right] \tag{10}$$

5. Overview

An overview of the SAC framework is shown in Fig. 1. With the critic loss and policy objective functions, a pseudocode can be constructed as shown in Algorithm 1.

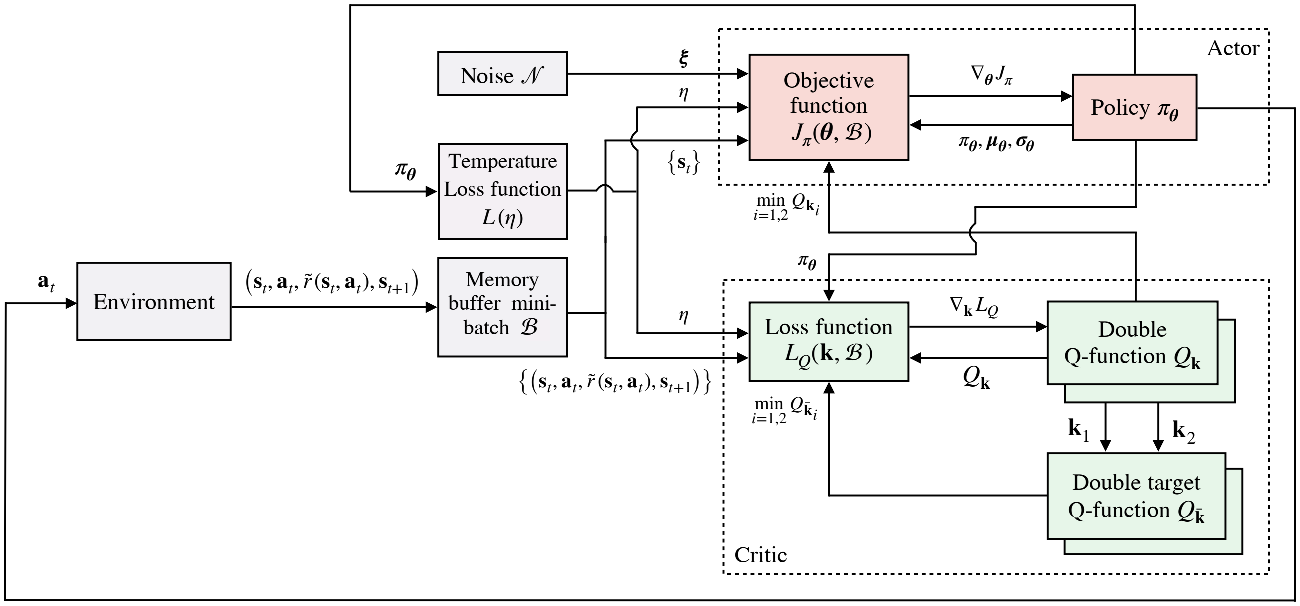

Actor
Noise $\mathcal{N}$
ξ
η
Objective function $J_\pi(\boldsymbol{\theta}, \mathcal{B})$
$\nabla_{\boldsymbol{\theta}} J_\pi$
Policy $\pi_{\boldsymbol{\theta}}$
$\pi_{\boldsymbol{\theta}}, \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{\boldsymbol{\theta}}$
$\pi_{\boldsymbol{\theta}}$
Temperature Loss function $L(\eta)$
$\{\mathbf{s}_t\}$
$\min_{i=1,2} Q_{\mathbf{k}_i}$
$\pi_{\boldsymbol{\theta}}$
$\mathbf{a}_t$
Environment
$(\mathbf{s}_t, \mathbf{a}_t, \tilde{r}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})$
Memory buffer mini-batch $\mathcal{B}$
η
Loss function $L_Q(\mathbf{k}, \mathcal{B})$
$\nabla_{\mathbf{k}} L_Q$
Double Q-function $Q_{\mathbf{k}}$
$Q_{\mathbf{k}}$
$\{(\mathbf{s}_t, \mathbf{a}_t, \tilde{r}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
$\min_{i=1,2} Q_{\bar{\mathbf{k}}_i}$
$\mathbf{k}_1$
$\mathbf{k}_2$
Double target Q-function $Q_{\bar{\mathbf{k}}}$
Critic

Fig. 1 SAC framework.

Algorithm 1: SAC. Adapted from [26].

```
Initialize $\boldsymbol{\theta}, \mathbf{k}_1, \mathbf{k}_2$ parameters for policy $\pi_{\boldsymbol{\theta}}$ and double Q-function $Q_{\mathbf{k}_{1,2}}$;
Set target parameters $\bar{\mathbf{k}}_1 \leftarrow \mathbf{k}_1$ and $\bar{\mathbf{k}}_2 \leftarrow \mathbf{k}_2$;
Initialize empty memory buffer $\mathcal{D}$, minibatch $\mathcal{B}$, learning rate $\lambda$ and smoothing factor $\tau$ ;
Observe initial state $\mathbf{s}_0$;
for each time step $t$ do
    Sample action $\mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \mathbf{s}_t)$ ;
    Observe next state and reward $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t), \tilde{r}(\mathbf{s}_t, \mathbf{a}_t)$ ;
    Store transition sample $(\mathbf{s}_t, \mathbf{a}_t, \tilde{r}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})$ in $\mathcal{D}$ ;
    Sample a minibatch of transition samples $\mathcal{B} = \{(\mathbf{s}_t, \mathbf{a}_t, \tilde{r}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ from $\mathcal{D}$ ;
    Update Q-function parameters: $\mathbf{k}_i \leftarrow \mathbf{k}_i - \lambda \nabla_{\mathbf{k}_i} \frac{1}{|\mathcal{B}|} \sum L_Q(\mathbf{k}_i, \mathcal{B})$ for $i = 1, 2$;
    Update policy parameter: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \lambda \nabla_{\boldsymbol{\theta}} \frac{1}{|\mathcal{B}|} \sum J_\pi(\boldsymbol{\theta}, \mathcal{B})$
    Update the temperature hyperparameter: $\eta \leftarrow \eta - \lambda \nabla_\eta L(\eta)$ ;
    Update target Q-function parameters: $\bar{\mathbf{k}}_i \leftarrow (1 - \tau)\mathbf{k}_i + \tau \bar{\mathbf{k}}_i$ for $i = 1, 2$;
end
```

III. Controller Design

With the SAC framework presented above, its integration with the flight controller is discussed in this section.

A. High-Fidelity Cessna Citation 500 Model

The system to be controlled in this application is a high-fidelity non-linear simulation model of the Cessna Citation 500 business jet aircraft. It was built with the Delft University Aircraft Simulation Model and Analysis Tool (DASMAT) based on flight data recorded on the PH-LAB research aircraft shown in Fig. 2. The model was validated in [27]. It is expected that the controller proposed in this research will be flight-tested on the PH-LAB at a later stage.

Fig. 2 Cessna Citation PH-LAB research aircraft.[*]

The full coupled-dynamics of the aircraft are to be controlled with a refresh rate of 100Hz. For this research, actuator dynamics are modeled with a low-pass filter and saturation limits, while ideal sensors are assumed. A yaw damper is already present on the aircraft. The aircraft is kept in a clean configuration for this simulation. The aircraft state and control input vectors are given in Eqs. (11) and (12), respectively. At the beginning of each simulation, the aircraft is untrimmed with null initial control inputs.

$$\mathbf{x} = [p, q, r, V, \alpha, \beta, \theta, \phi, \psi, h]^\intercal \tag{11}$$

$$\mathbf{u} = [\delta_\mathrm{e}, \delta_\mathrm{a}, \delta_\mathrm{r}]^\intercal \tag{12}$$

B. Interfacing

In reinforcement learning, the flight controller, the plant and the control input are known as the agent, the environment and the action, respectively.

A flight controller for automatic altitude and attitude tracking is to be built as part of this research so that it can be interfaced with existing navigation algorithms reviewed in [28]. To ensure compatibility with the experimental fly-by-wire system of the PH-LAB in future flight tests, the controller developed in this research will only command control surfaces while the airspeed is controlled with an independent PID auto-throttle. A flight control task for altitude, roll and sideslip angles tracking is proposed. Because of the difference in their dynamics, a more stable learning is expected by having the altitude and attitude controlled by two separate, cascaded controllers.

1. Attitude Control

An inner-loop SAC agent will track reference signals for the pitch, roll and sideslip angles, referred to with the R superscript. A reward function based on the clipped L1 norm of the error vector is proposed in Eq. (15). A cost vector $\mathbf{c}$, determined by trial and error, is associated with the tracked states in Eq. (14), where the sideslip angle is attributed a higher cost due to its generally low magnitude.

$$\mathbf{e}^{\mathrm{att}} = \left[\beta^R - \beta, \theta^R - \theta, \phi^R - \phi\right]^\intercal \tag{13}$$

$$\mathbf{c}^{\mathrm{att}} = \frac{6}{\pi}\,[4, 1, 1]^\intercal \tag{14}$$

$$\tilde{r}(\mathbf{e}^{\mathrm{att}}) = -\frac{1}{3}\left\|\mathrm{clip}\left[\mathbf{c}^{\mathrm{att}} \odot \mathbf{e}^{\mathrm{att}}, \vec{\mathbf{-1}}, \vec{\mathbf{0}}\right]\right\|_1 \tag{15}$$

Initial tests found that the aircraft control input was noisy when corresponding directly to the agent action. To smooth out the control input, the agent is set to command the control input increment $\mathbf{\Delta u}$ in Eq. (16). Because the tanh function squashes the action vector $\mathbf{a}^{\mathrm{att}}$ to $[-1, 1]^3$, it is mapped to the physical range of the control input increment in Eq. (17), chosen as a hundredth of actuator limits to prevent sharp variations.

$$\mathbf{u}_t = \mathbf{u}_{t-1} + \mathbf{\Delta u}_t \tag{16}$$

$$\mathbf{\Delta u} = \mathbf{\Delta u}_{\mathrm{min}} + (\mathbf{a}^{\mathrm{att}} + 1)\frac{\mathbf{\Delta u}_{\mathrm{max}} - \mathbf{\Delta u}_{\mathrm{min}}}{2} \tag{17}$$

A trade-off has to be made between making the environment state smaller to speed up learning and giving enough information to allow the agent to make informed decisions. As the environment is assumed to have the Markov property, only information of the current time step is needed to explain its state. The environment state is decided to contain the weighted error vector to ensure a satisfactory steady-state response, the three body rates to improve the transient

[*] Image from C. v. Grinsven (with permission).

response, and the current control input since the agent only controls its increment. The environment state is given in Eq. (18).

$$\mathbf{s}^{\text{att}} = \left[\left(\mathbf{c}^{\text{att}} \odot \mathbf{e}^{\text{att}}\right)^\intercal, \mathbf{u}^\intercal, p, q, r\right]^\intercal \tag{18}$$

2. Altitude Control

An outer-loop SAC agent is tasked with providing a reference pitch angle to track an altitude signal. The error and cost vectors are defined in Eqs. (19) and (20), respectively. Similarly to the inner-loop agent, the reward function is defined as the absolute clipped weighted error in Eq. (21). The cost was found empirically so that, despite the clipping, differences between large errors are perceived by the agent, while still incentivizing for a low steady-state error.

$$\mathbf{e}^{\text{alt}} = \left[h^R - h\right] = [\Delta h] \quad (19) \qquad \mathbf{c}^{\text{alt}} = \left[\frac{1}{240}\right] \quad (20) \qquad \tilde{r}(\mathbf{e}^{\text{alt}}) = -\left\|\text{clip}\left[\mathbf{c}^{\text{alt}} \odot \mathbf{e}^{\text{alt}}, \vec{\mathbf{-1}}, \vec{\mathbf{0}}\right]\right\|_1 \quad (21)$$

For this agent too, the control policy is smoothed by controlling the reference pitch angle increment instead of its value in (22). The agent action $\mathbf{a}^{\text{alt}}$ defined on $[-1, 1]$ is mapped to the pitch angle increment in Eq. (23) such that the corresponding pitch rate does not exceed 10deg s^{-1}.

$$\theta^R_t = \theta^R_{t-1} + \Delta\theta^R_t \quad (22) \qquad \Delta\theta^R = \Delta\theta^R_{\min} + (\mathbf{a}^{\text{alt}} + 1)\frac{\Delta\theta^R_{\max} - \Delta\theta^R_{\min}}{2} \quad (23)$$

The state environment in Eq. (24) is reduced because the agent only has to learn the kinematic relationship between altitude and pitch angle. No knowledge of lateral states is needed since the inner-loop controller is already fully coupled.

$$\mathbf{s}^{\text{alt}} = \left[\mathbf{c}^{\text{alt}} \odot \mathbf{e}^{\text{alt}}, \theta^R\right]^\intercal \tag{24}$$

A diagram of the cascaded controller structure is shown in Fig. 3. Feedback signals of plant state variables, current pitch reference angle and control surface deflections are from the previous time step.

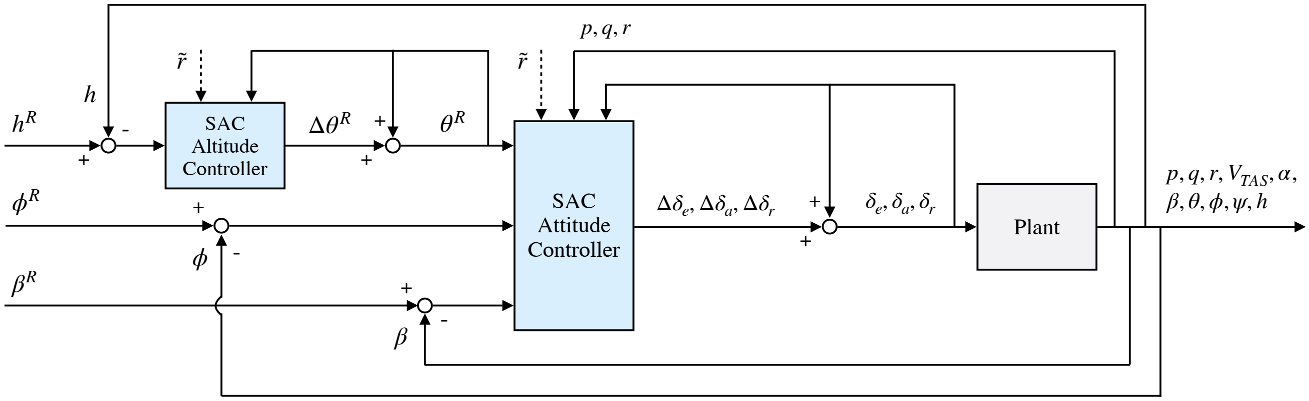


Fig. 3 Cascaded controller structure for altitude and attitude control. The SAC controllers observe the weighted errors, untracked states, and current control input and also receive a reward from the environment.

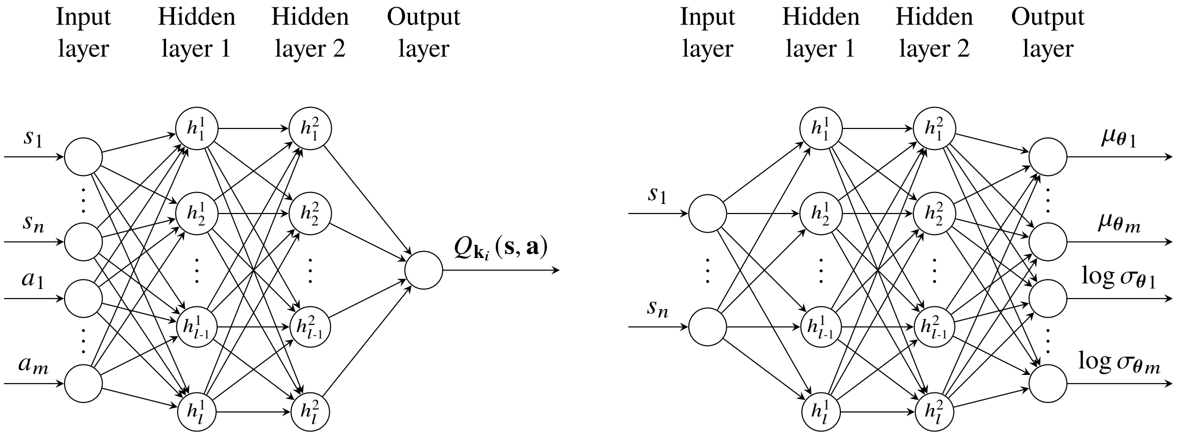
3. Hyperparameters

Choosing suitable hyperparameters for the SAC controllers and DNNs can significantly improve sample efficiency. A short selective-search hyperparameter optimization centered on default values was performed on the learning rate, number of hidden units, memory buffer and minibatch sizes, all known to be influential hyperparameters. In particular, it was found that the altitude controller, with smaller state and action spaces, performed better with fewer hidden units

Table 1 Cascaded SAC controller hyperparameters. Default values from [26].

Hyperparameter	**Altitude Controller**	**Attitude Controller** (if different)
Learning rate λ	$3 \cdot 10^{-4}$	Linearly decreasing from $4 \cdot 10^{-4}$ to 0
Hidden units lxl	32x32	64x64
Entropy target $\log \bar{\mathcal{H}}$	$-m = -1$ (default)	$-m = -3$ (default)
Discount factor γ	0.99 (default)	
Network activation	ReLu (default)	
Memory buffer size $\lvert\mathcal{D}\rvert$	$5 \cdot 10^4$	
Minibatch size $\lvert\mathcal{B}\rvert$	256 (default)	
Smoothing factor τ	0.995 (default)	

and a higher average learning rate. Several default hyperparameters are used from the original SAC implementation [26]. An overview of chosen hyperparameters is presented in Table. 1. Additionally, network initialization is executed following the Xavier method [29], and the gradient-descent employs the Adam optimizer, recognized for its superiority in terms of learning stability [30].



(a) Q-function i={1,2}

(b) Policy

Fig. 4 Network topology of SAC controllers. The altitude controller has $n = 2$, $m = 1$ and $l = 32$ while the attitude controller has $n = 9$, $m = 3$ and $l = 64$. Subscript j of s_j represents the j-th element of vector s.

The network topology for the controllers is visualized in Fig. 4. The double Q-function corresponds to two independent networks with the structure shown in Fig. 4a. The policy's multivariate Gaussian distribution is defined according to the mean and log-standard deviation vectors from the network output in Fig. 4b. Using log-standard deviations allows the network to estimate the parameter on $\mathbb{R}$ and exponentiation is used to sample actions from the policy according to Eq. (8). At evaluation time, actions are made deterministic for consistent performance by setting the noise vector in Eq. (8) to zero.

Hidden units contain a linear combination of the input vector with bias and are fed to a normalization layer implemented according to [31]. This is to reduce the difficulty of training networks whose inputs have different scales and non-zero means. The normalized value is then given to a ReLu activation function, as seen in Fig. 5 for a generic hidden unit. The output layers, on the other hand, only contain a linear combination of the second hidden layer units with bias. All weights, biases and normalization factors are contained in network parameter vectors **k** and $\boldsymbol{\theta}$ for the Q-function and the policy, respectively.

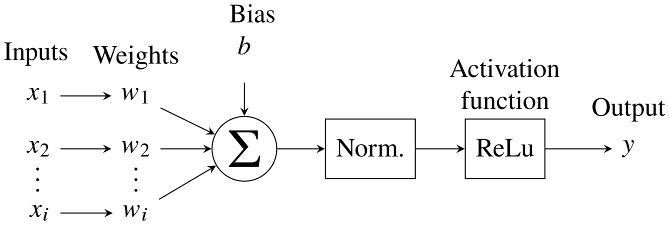


Fig. 5 Hidden unit h with input vector x and scalar output y.

C. Experiment Setup

In this research, the SAC controllers are trained offline on the normal plant dynamics and their adaptation to failures is subsequently evaluated online based on their robust response. For comparison purposes, the adaptive response to failures is also generated.

1. Offline Learning

DRL agent training is best performed offline as it typically requires 10^6 time steps or more. Normal plant dynamics with initial altitude and speed of 2000m and 90ms^{-1}, respectively, are used for the entire training process. For better learning stability, the inner-loop attitude controller is trained first alone with step reference signals for the pitch and roll angles, while the sideslip reference is always zero. After 500 20s-training episodes, or 10^6 time steps, the positive learning curve in Fig. 6 reaches a plateau, suggesting that no further training is required.

The altitude controller training is subsequently performed with the fully trained attitude controller in the inner loop. Successive climbing, constant altitude, and descending tasks allow the controller reach a converged policy after 10^6 time steps. As observed in Fig. 6, training is quite unstable as significant performance drops are experienced even in the last training stages. Benefiting from the offline learning environment, training for both controllers is repeated until a satisfactory policy is reached.

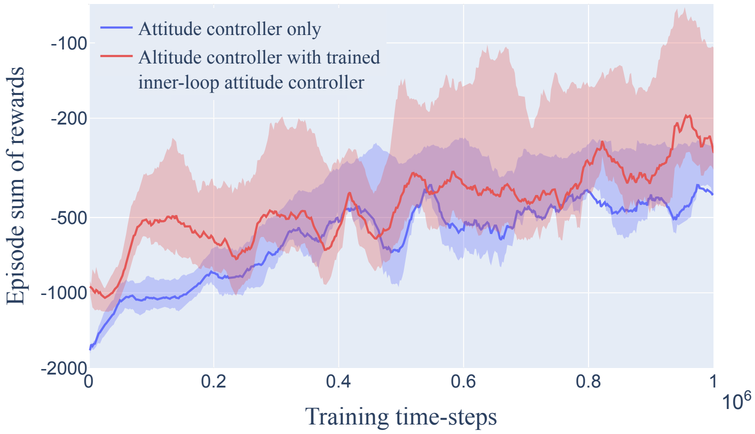


Fig. 6 Sum of rewards in function of training time-steps. The curves show the mean (smoothed with a window of length 20) and the shaded region the interquartile range over 5 random seeds of successful trials.

2. Online Robust Adaptation

The adaptation of the cascaded SAC controller to unseen flight conditions, atmospheric disturbances and sensor noise is evaluated with its robust response, as no controller parameters are changed. Similarly, six unknown failure cases are simulated online and the controller has to robustly adapt its response without changing its parameters.

As an additional experiment, the SAC attitude controller is also trained on the failed system. While this experiment falls outside the scope of the research objective since simulation models of failed dynamics are typically not obtainable, it provides insights into possible performance improvements. The previously-described offline training process of the attitude controller is repeated for each failure case to obtain six adaptive agents. The adaptive response will, however, not be used to assess the fault-tolerance of the SAC controller.

IV. Results and Discussion

The Cessna Citation jet aircraft response with the SAC controller is evaluated in this section, both on the non-failed and failed system. Moreover, the effect of biased sensor noise and atmospheric disturbances on the response is assessed. Lastly, additional robustness and reliability tests are conducted.

A. Non-Failed System

One of the goals of this experiment is to show that the controller can operate the non-failed aircraft on a representative coupled attitude tracking task. As depicted in Fig. 7, the attitude controller alone is able to keep the aircraft stable and minimizes the error overall. As the aircraft is initially untrimmed, rapid variations in the control input is seen close to $t = 0$s. A large roll angle is reached despite remaining 7.5°-off from the 70° reference, as the controller finds a compromise with the pitch angle error given the difficulty of keeping the aircraft horizontal during such a large bank maneuver. The roll angle suffers from a small yet consistently positive steady-state error, which could be due to the controller having failed to completely learn the asymmetricity of the aircraft.

The response of a similar aircraft on an analogous roll angle task in [5] with a model-based DHP controller shows less transient-response damping and a sideslip angle up to 50 times larger. The velocity, on the other hand, is better tracked there since it is directly controlled by the DHP agent, unlike here with an external auto-throttle.

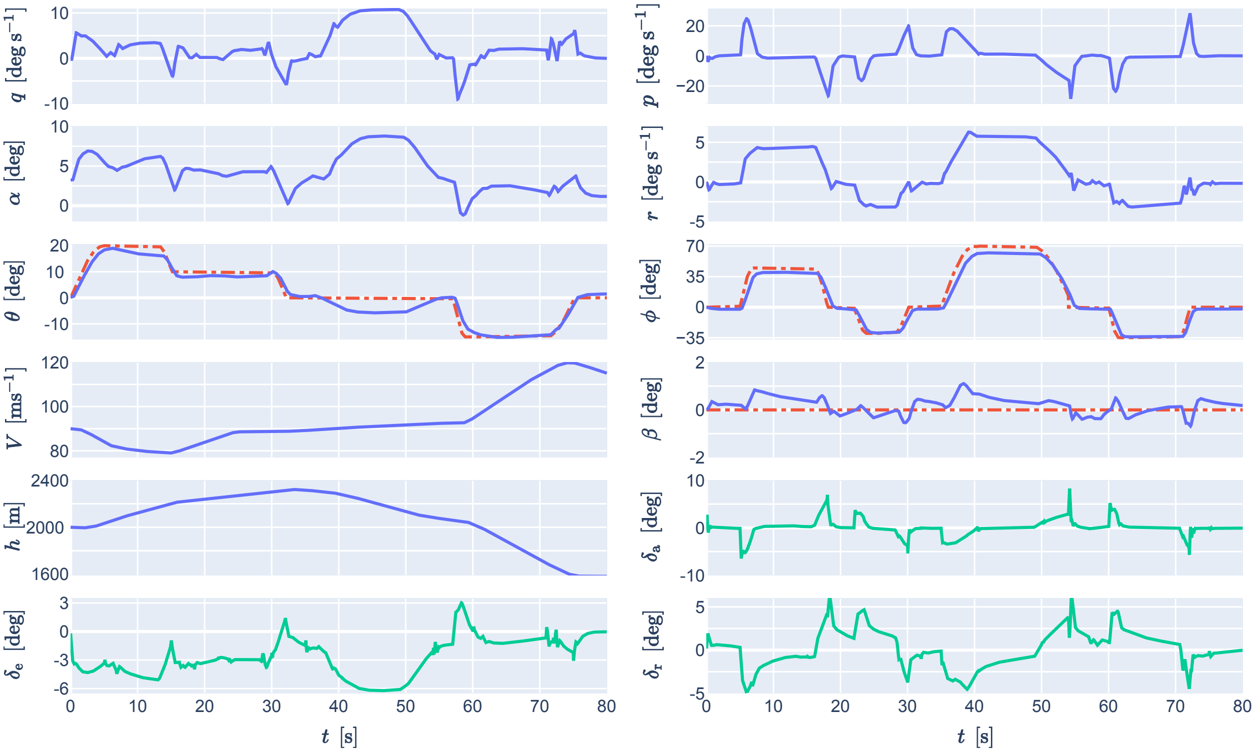


Fig. 7 Attitude tracking response with SAC attitude controller. Reference signals are shown with red dashed lines and control inputs with green solid lines.

The proposed framework should also be able to complete a representative altitude tracking task with the cascaded controller structure. As seen in Fig. 8, successive climbing turns are tracked well, with an altitude error of 15m or lower. This confirms that expert knowledge in flight control is successfully used to integrate the inner-loop and outer-loop controllers. Coupling effects in the attitude controller are observed as the elevator is pushed further up during the 40° bank turns at $t = 35$s and $t = 65$s, which is not sufficient to track the altitude controller's higher pitch angle reference. This could be due to the attitude controller avoiding regions of pitch rate higher than 5°/s as long as the pitch error is not too large, unlike in Fig. 7 where it reached 6°.

Similar performance to the Cessna Citation 500 response with uncoupled IDHP agent in [10] is achieved, this time

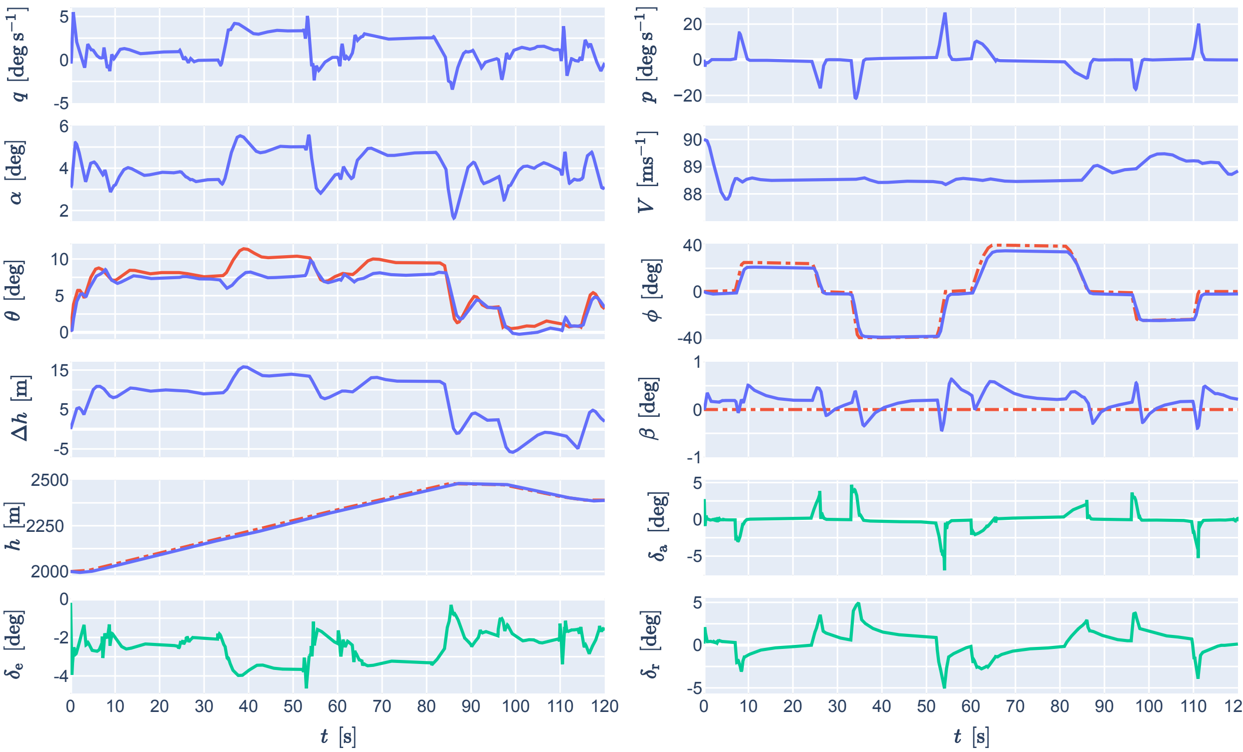


Fig. 8 Altitude tracking response with cascaded SAC controller. External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines.

with a maximum roll angle of 40° instead of 25°, benefiting from the coupled-dynamics SAC attitude controller.

B. Failed System

Another goal of this research is to evaluate the cascaded SAC controller on several types of failures. The robust response is shown first as it experiences an unexpected change in plant dynamics not seen during training. To identify possible performance gains, it is followed by the adaptive response, corresponding to a SAC controller trained on the failure case. For this research, however, the robust response is most important to determine if the SAC controller is successful since a plant model for every failure is typically not available for offline training.

1. Jammed Rudder

The response on a failure case with the rudder struck at $\delta_r = -15°$ from $t = 10$s onward is displayed in Fig. 9. The robust response until $t = 60$s is stable despite the severe failure. The sideslip error is inevitably big with this rudder deflection, but large and relatively constant 15° and 5° errors in the roll and pitch angles, respectively, are also observed. The attitude controller is affected by the unusual dynamics, resulting in large body rates oscillations at the failure time. They are nevertheless contained as it starts operating the elevator and ailerons around a new offset position, and as expected, it also tries to deflect the rudder in the opposite direction to the one it is stuck in, with no effect. The altitude controller, on the other hand, manages to track the climb task successfully by producing a higher-than-normal pitch reference, thereby almost removing the effect of the large pitch angle error. Overall, the robust controller leverages its knowledge of coupling effects jointly using the elevator and ailerons around a new offset to counteract the rudder failure.

The adaptive response was obtained by training the controller on the failed system and by stopping it both from tracking the sideslip and controlling the jammed rudder. The response then exhibits a tracking performance similar to the non-failed case in the pitch and roll angles even though the sideslip angle error is still large.

A similar failure case on a business-jet aircraft with a model-dependent DHP controller with the rudder stuck at $\delta_r = -15°$ and an asymmetric loss of thrust (possibly beneficial for this failure case) was investigated in [5]. The adaptive response was stable but showed large 10°-amplitude undamped sideslip angle oscillations and a roll angle error of up to 50°. Oscillations in the longitudinal plane had five times higher amplitude and two times higher frequency than the robust controller in Fig. 9. Overall, the SAC robust response is more stable with smaller oscillations and lower error than the DHP adaptive one. This is partly attributable to the high generalization power of DNNs and the robustness of stochastic policies.

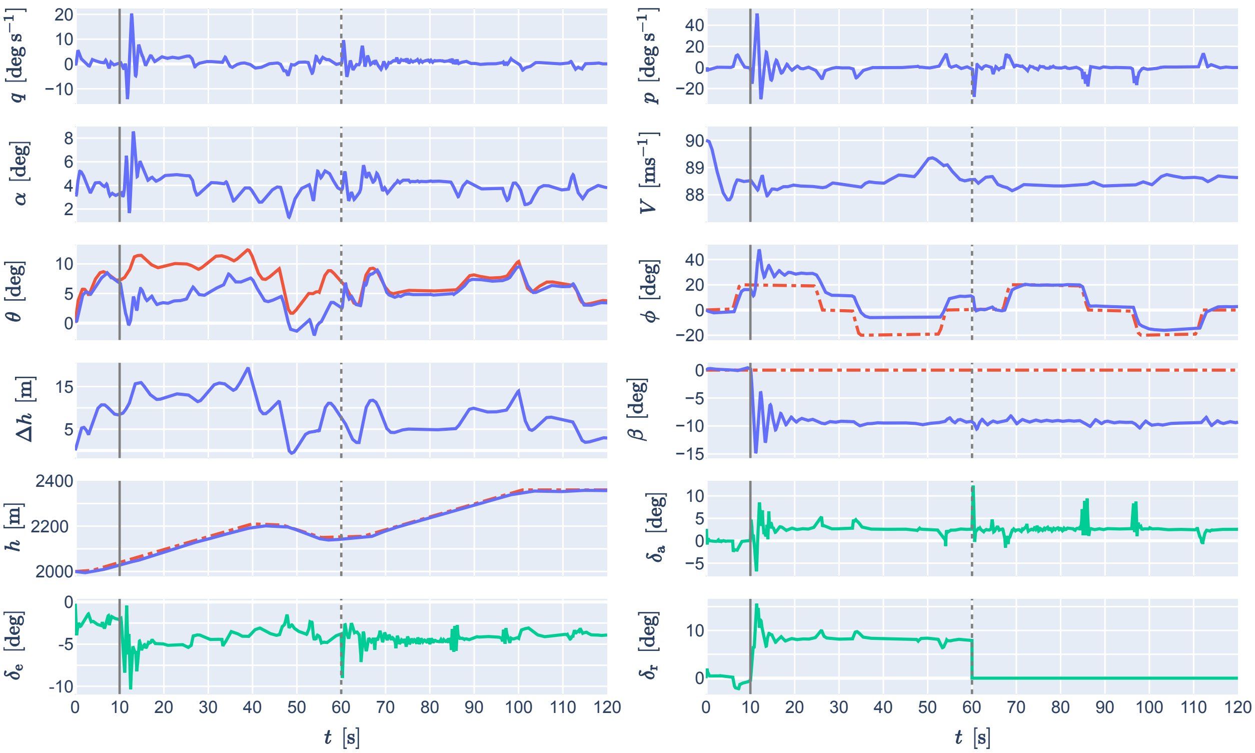


Fig. 9 Altitude tracking response with rudder stuck at $\delta_r = -15°$ from $t = 10$s (grey solid line). External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines. Robust control until $t = 60$s, adaptive control thereafter (grey dotted line).

2. Reduced Aileron Effectiveness

An aileron failure case, implemented as a 70% reduced aileron effectiveness, is depicted in Fig. 10. The aileron failure has few effects on the response. Most notably, the robust controller more than doubles and prolongs non-zero aileron deflections to track the roll angle reference with almost no difference compared to the non-failed plant. The roll rate decreases by up to 6% with respect to the non-failed value.

The adaptive response from $t = 90$s exhibits almost identical performance, although its control policy is reduced in magnitude resulting in a slightly slower transient response in the roll angle tracking. It also suffers from some oscillations in the longitudinal plane, making its response less desirable than the robust controller. The lower performance likely comes from the increased difficulty to learn on the failed system.

In [10], an aileron failure with a 50% reduced effectiveness was introduced on a Cessna Citation 500 with an IDHP controller. Benefiting from its adaptive capability, the response was stable and well tracked in spite of the failure. It is worth noting that the robust SAC controller showed equivalent performance on a more severe failure, indicating again the high robustness of stochastic policies.

3. Reduced Elevator Range

Another actuator failure is tested, this time with the elevator range reduced from [−20.05°, 14.90°] to [−2.50°, 2.50°]. The SAC agent is unaware of the failure and control inputs outside the new range are saturated. The robust response

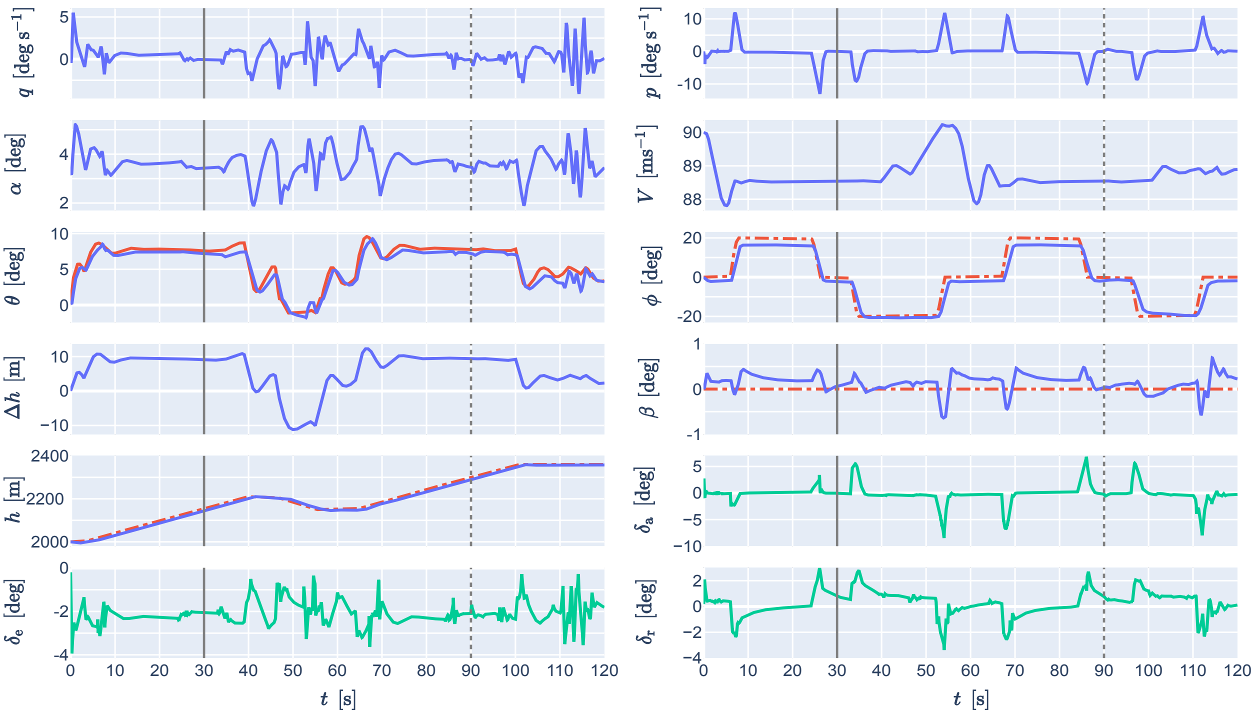


Fig. 10 Altitude tracking response with 70% reduced aileron effectiveness from t = 30s (solid grey line). External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines. Robust control until t = 90s, adaptive control thereafter (grey dotted line).

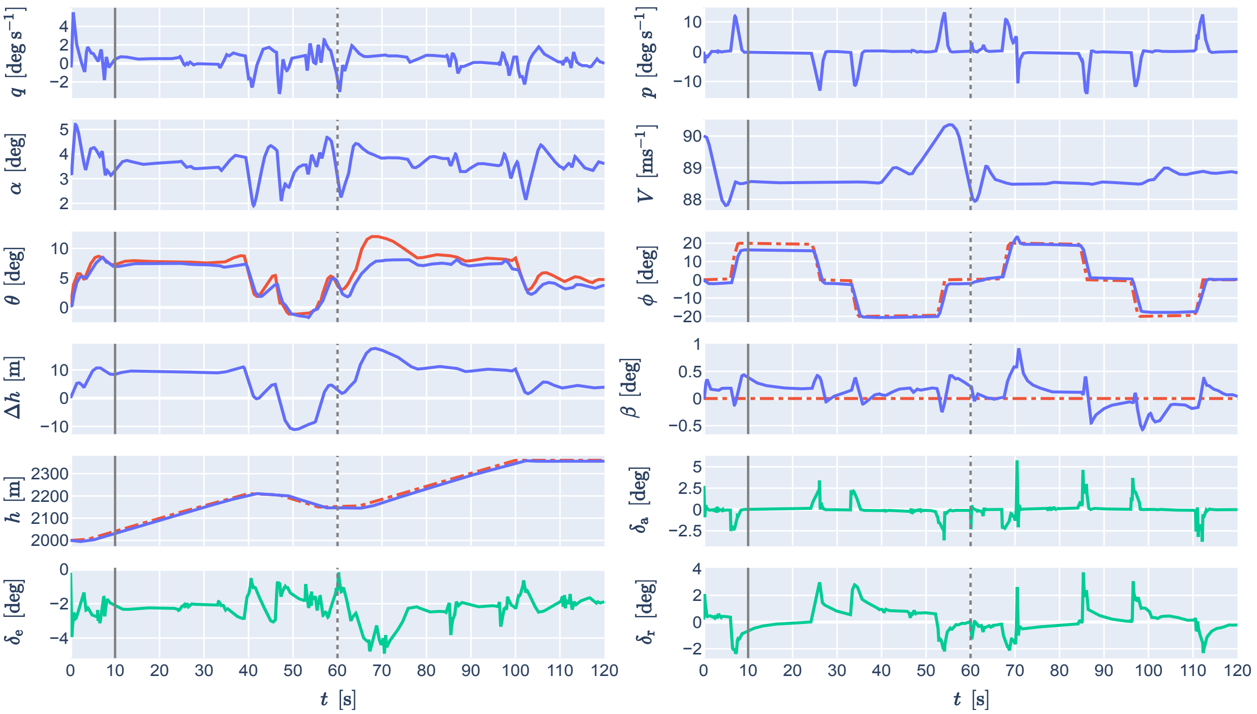


Fig. 11 Altitude tracking response with reduced elevator range from t = 10s (grey solid line). External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines. Robust control until t = 60s, adaptive control thereafter (grey dotted line).

shown in Figure 11 is little-affected by the failure from t = 10s. While all states remain stable and well-tracked, the maximum pitch rate is reduced and remains at about half of the non-failed case seen in Fig. 8. This is explained by the close relationship between elevator deflection and pitch rate.

The adaptive response, from t = 60s, degrades more because of the failure. Despite having been trained on this failure, it can be explained by the more difficult task of initiating the climb with the failure already present. The controller instructs for more than 10s a control input outside the reduced elevator deflection range, which is not enough to avoid a large pitch error. The altitude error, in turn, doubles with respect to the non-failed case but the aircraft eventually completes the climb. It is worth noting that the agent did not explicitly learn the new saturation limits as its control input exceeds the bounds.

4. Partial Loss of Horizontal Tail

Structural failures are difficult to anticipate as they can affect various components of the aircraft. For this study, the structural failure of an essential part to flight control, the horizontal tail, is studied. As it mainly affects the elevator control effectiveness and pitch damping, it is implemented in the simulation model with a 70% reduction in $C_{(L/D/m)_{\delta_e}}$ and C_{m_q}.

The failure at t = 10s generates a sudden loss in elevator effectiveness, which the robust agent immediately counteracts by quadrupling the elevator deflection. Despite that, large pitch angle and altitude errors are observed as the achieved pitch rate is too low for the climbing task. Lateral motion states are unaffected by this failure. This failure case shows the ability of the agent to adapt to this unexpected structural failure by offsetting its control input.

The adaptive response from t = 60s shows a similar control strategy with an unusually high elevator deflection while keeping most states well-tracked.

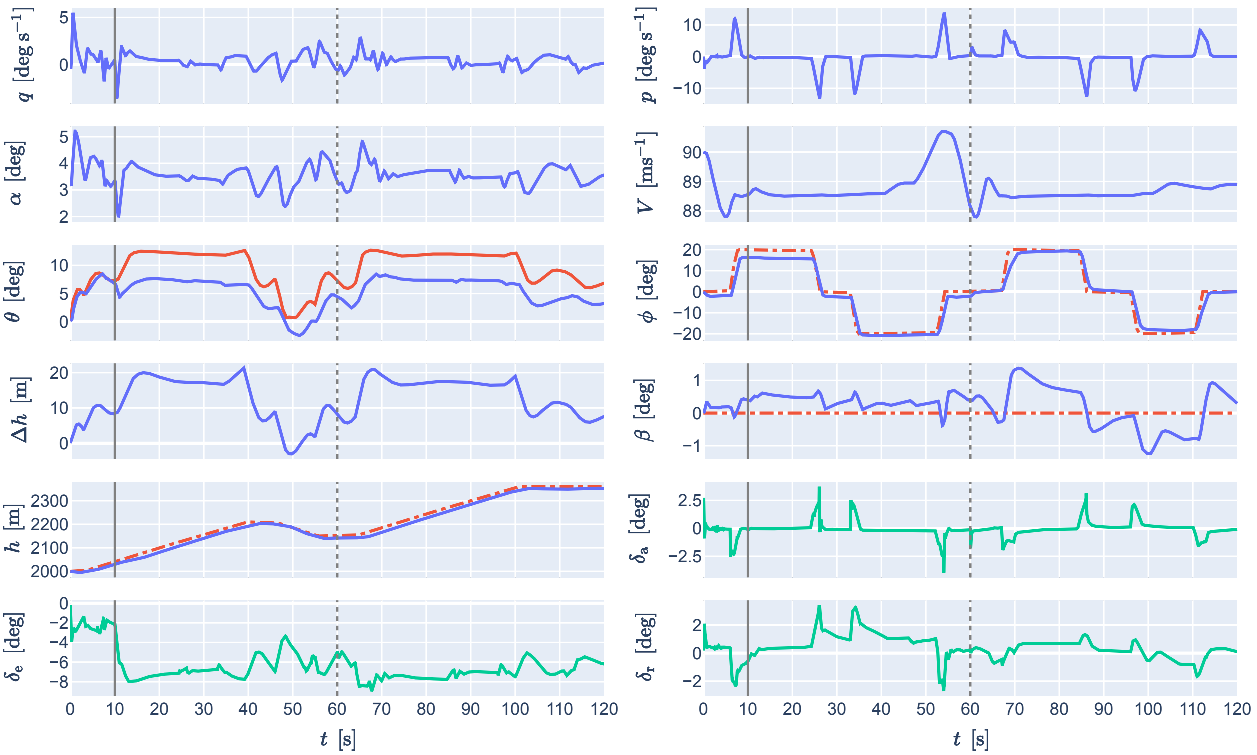


Fig. 12 Altitude tracking response with partial loss of horizontal tail from t = 10s (grey solid line). External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines. Robust control until t = 60s, adaptive control thereafter (grey dotted line).

5. Icing

Icing is a common challenge faced by aircraft as ice accretions accumulate on wings, which main effects include a reduction in maximum lift coefficient and an increase in drag coefficient [32]. Conservative estimates suggest a reduction of $C_{L_{\max}}$ by 30% for mid-range Reynolds numbers and an increase of C_D by 0.06. As observed in Fig. 13, the decrease in lift and increase in drag due to icing cause large positive pitch and 20m-altitude errors on the robust response. The robust controller still manages to achieve the climbing task by doubling its elevator deflection, although it avoids deflecting it further up than $-4°$, most likely to avoid the stall region. The auto-throttle is unable to deal with both the increased drag and the climbing task, which leads to a reduction in velocity of 13% compared to the non-iced plant. Lateral states, on the other hand, are unaltered by icing.

The adaptive controller takes a more aggressive control strategy by deflecting the elevator further up, and despite trading kinetic energy, generates a larger altitude error. The roll angle transient response deteriorates and the sideslip error increases compared to the robust controller. It seems that the learning task with icing was difficult for the controller, which converged to a worse policy than the one of the robust controller.

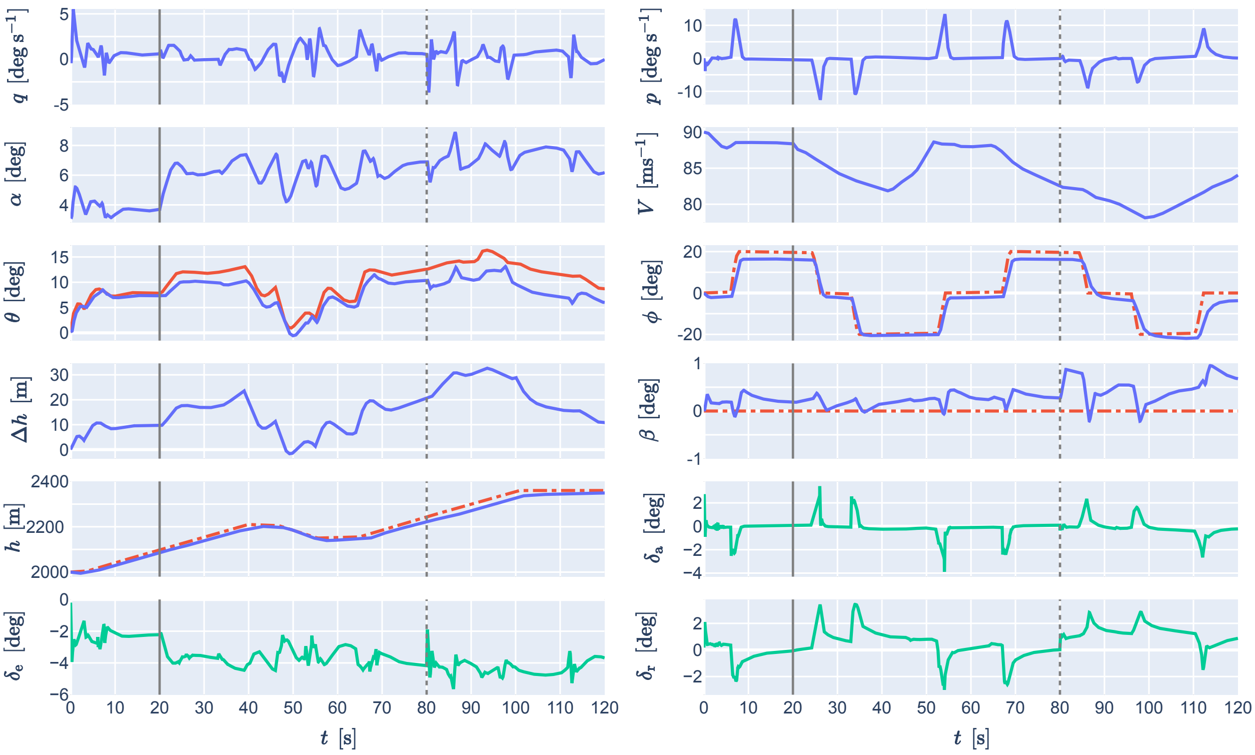


Fig. 13 Altitude tracking response with icing from $t = 20$s (solid grey line). External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines. Robust control until $t = 80$s, adaptive control thereafter (grey dotted line).

6. Center-of-Gravity Shift

A sudden backward shift of heavy cargo in the aircraft can cause a diminution of the stability margin and create instability. The event of a 300kg payload moving from the front to the back of the passenger cabin is investigated, which translates to a backwards c.g. shift of 0.25m on the PH-LAB. As seen in Fig. 14, the c.g. shift at $t = 20$s causes a sudden change in the pitch rate and the controller immediately adapts the elevator deflection to the rare positive range. A constant negative pitch angle error appears as the c.g. moves backward, indicating that the controller's policy is stable but has not fully adapted to the new c.g. location. This is because the controller has only knowledge of the pitch error and not of the pitch angle itself, and has likely attributed higher elevator deflections to steep dive maneuvers on the aircraft with normal c.g. position. The robust attitude controller is unable to offset its elevator control input accurately. Consequently, the aircraft now has difficulty pitching down and experiences a large negative altitude error during descent. The lateral states, on the other hand, remain undisturbed by the c.g. shift.

The adaptive controller eliminates most of the pitch angle error, although it still suffers from sudden pitch-up events as the c.g. moved further back. This hints at instability issues should the c.g. be moved further back.

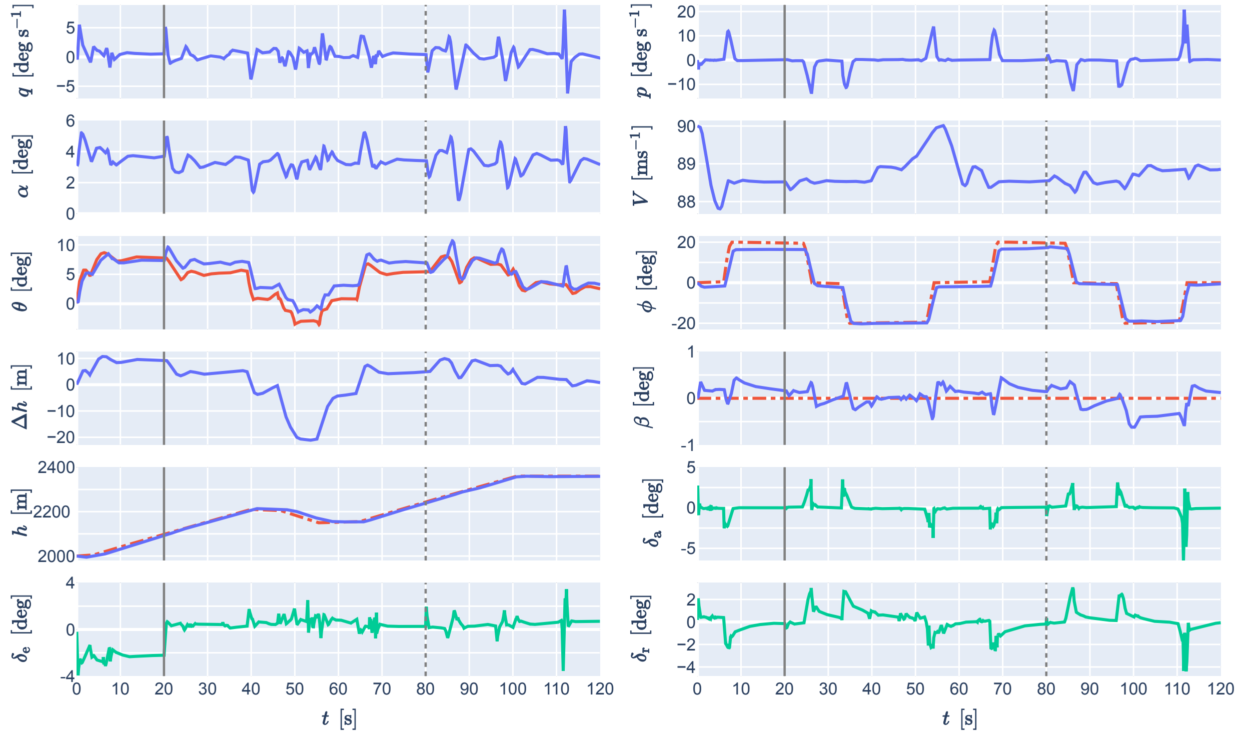


Fig. 14 Altitude tracking response with c.g. shift at t = 20s (solid grey line). External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines. Robust control until t = 80s, adaptive control thereafter (grey dotted line).

C. Effect of Biased Sensor Noise and Atmospheric Disturbances

To evaluate the SAC controller in conditions closer to real-world phenomena, the addition of biased sensor noise and atmospheric disturbances is studied. A Gaussian white noise with standard deviation and bias obtained from the Cessna Citation PH-LAB's sensors in [33] are used to simulate sensor noise, with values shown in Table. 2. Noise from control surface deflection measurements is disregarded as the attitude controller observes the control input instead. Atmospheric disturbances in the form of discrete vertical gusts (15ft s^{-1}), specified in MIL-F-8785C [34], are also added to the system. They are implemented as 3s-step disturbances on the angle-of-attack at t = 20s and t = 75s.

Table 2 Cessna Citation PH-LAB aircraft sensor characteristics. Values from [33].

Observed state	p, q, r **[rad s^{-1}]**	θ, ϕ **[rad]**	β **[rad]**	h **[m]**
Noise SSD	$6.3 \cdot 10^{-4}$	$3.2 \cdot 10^{-5}$	$2.7 \cdot 10^{-4}$	$6.7 \cdot 10^{-2}$
Bias	$3.0 \cdot 10^{-5}$	$4.0 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$8.0 \cdot 10^{-3}$

As shown in Fig. 15, despite the controllers only having been trained with the ideal sensor assumption, the response remains stable and the errors are similar to the ideal-sensor scenario. The addition of biased sensor noise creates a slightly noisy control input and reference pitch angle while the aircraft states are unaffected.

The upwards vertical gusts cause a sudden increase in the angle-of-attack and disturb the pitch rate. As this makes the altitude error magnitude decrease, the outer-loop controller can instruct a lower pitch attitude, which subsequently leads to a downward elevator deflection. Conversely, when the gust vanishes, the sudden drop in angle-of-attack leads to a sharp upward elevator deflection to limit the altitude error. During this time, the lateral states are unperturbed. This response demonstrates the ability to reject atmospheric disturbances in the presence of biased sensor noise.

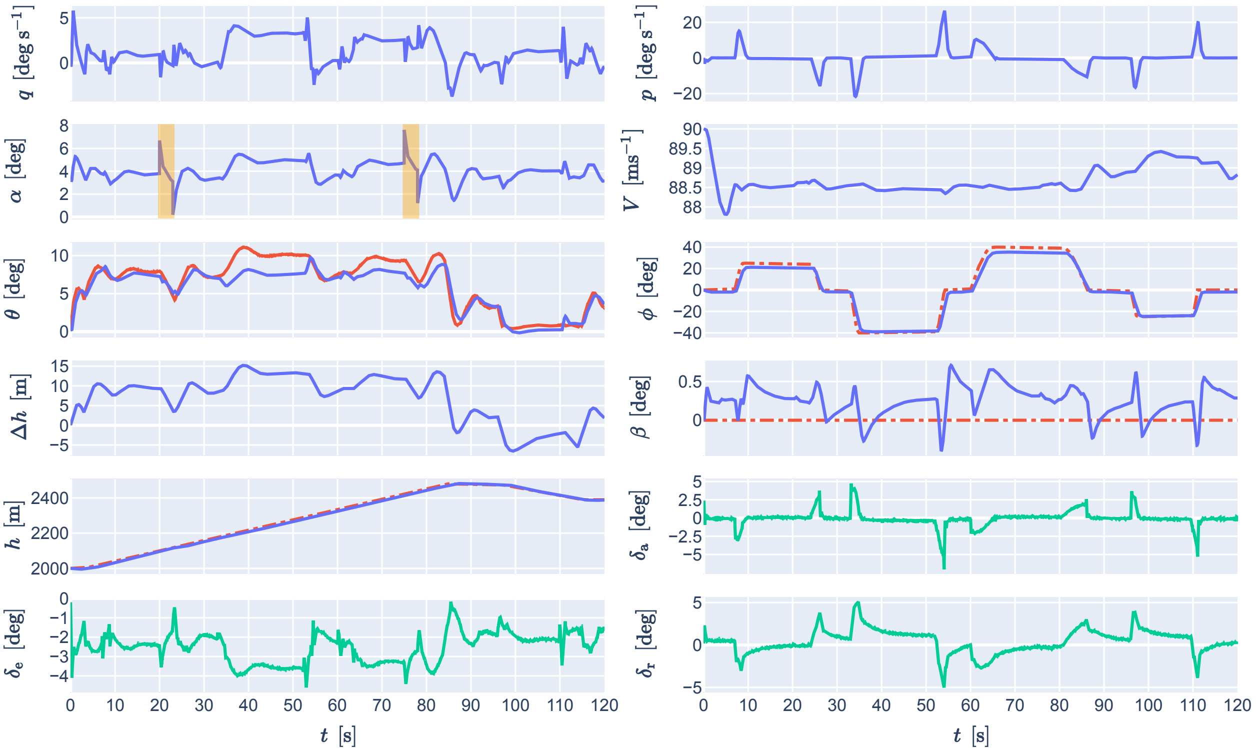


Fig. 15 Altitude tracking response with biased sensor noise and 15ft s^{-1} discrete vertical gusts (orange shaded region on α plot). External and self-generated reference signals are shown with red dashed and solid lines, respectively, and control inputs with green solid lines.

D. Additional Tests

Further tests in terms of robustness and training reliability are conducted in this section, in an effort to explore some limitations of the SAC controller.

1. Robustness Analysis

The cascaded SAC controller was evaluated so far on the same nominal Initial Flight Conditions (IFC) and similar reference signal shapes. It is interesting to investigate how general the controller is by evaluating its robustness to IFC it has not been trained on. Initial altitude and speed combinations are chosen to match previous tests done in [10, 12] on the same aircraft. The robustness to various reference signal shapes and frequencies is also evaluated, namely on sinusoidal and triangular reference signals for the altitude and roll angle reference signals, while the sideslip reference angle remains zero. A performance metric is established as the average normalized Mean Absolute Error (nMAE) over externally tracked states (altitude, roll and sideslip angles). Normalization is done over the reference signals range, while for the zero-referenced sideslip angle an acceptable range of $[-5°, 5°]$ is used.

As reported in Table 3, the controller is robust on all four IFC, where the largest error corresponds to the condition with the lowest dynamic pressure. A higher flight speed leads to an increased performance while a higher altitude leads to a decreased one, explained by higher and lower dynamic pressure and aerodynamic damping, respectively. Different reference signal shapes are tracked with a normalized Mean Absolute Error (nMAE) no higher than 5%, with the error mostly caused by the slower transient response of altitude on the high-frequency signals. Overall, the controller is observed to maintain a relatively low error on various IFC and a multitude of reference signal shapes.

Table 3 Robustness analysis to varying IFC and reference signal shapes. The controller was trained on the nominal IFC.

Reference signals h^R, ϕ^R	Initial altitude [m]	Initial speed [ms^{-1}]	nMAE
As in Fig. 8	2000 (nominal)	90 (nominal)	2.64%
As in Fig. 8	2000	140	1.95%
As in Fig. 8*	5000	90	3.16%
As in Fig. 8*	5000	140	2.13%
Sinusoidal (low frequency)	2000	90	3.22%
Sinusoidal (high frequency)	2000	90	5.00%
Triangular (low frequency)	2000	90	3.00%
Triangular (high frequency)	2000	90	4.76%

low frequency: $T_{h^R} = 80$s, $A_{h^R} = 80$m, $T_{\phi^R} = 50$s, $A_{\phi^R} = 50°$

high frequency: $T_{h^R} = 40$s, $A_{h^R} = 40$m, $T_{\phi^R} = 25$s, $A_{\phi^R} = 25°$

*h^R is offset to new initial altitude

2. Training Reliability Analysis

Training the SAC controllers involves several random processes, including sampling from the stochastic policy to choose actions, parameter initialization for the DNNs and minibatch sampling. Since the performance can vary from training to training, it is interesting to evaluate how reliable the training process is. Although a large enough number of trials can be granted in an offline learning environment, a high success rate would indicate the convenience of such process.

Samples are generated by training pairs of cascaded SAC controllers in the same conditions as described in subsubsection III.C.1. Each sample is evaluated on the nominal IFC and altitude tracking task shown in Fig. 8 with a 5% nMAE success threshold. A global training success rate is obtained by generating enough samples ($n = 27$) until convergence to a stable success rate is obtained. It is found to be at 26%. This low value reveals the difficulty to train with consistency stochastic policy-based and random initialization-dependent DRL algorithms. The high sample efficiency, generalization power and robustness to failures presented so far can be seen to come at the expense of learning stability. It should be noted, however, that this does not compromise the online reliability and performance of the SAC controller, as long as it is trained in an offline environment.

V. Conclusion

A controller employing Soft Actor-Critic (SAC) Deep Reinforcement Learning (DRL) for the coupled-dynamics control of the Cessna Citation 500 jet aircraft is built as part of this research. Expert knowledge from classic flight control is used to create a cascaded SAC controller structure. The response is stable and keeps the tracking error low to successfully achieve coordinated 40°-bank climbing turns and 70°-bank flat turns. Severe failures such as the rudder jammed at $-15°$, the aileron effectiveness reduced by 70% or other unforeseen events such as icing and c.g. shift are handled by the robust SAC controller such that stability is maintained and the tracking task is achieved, demonstrating the SAC controller's strong fault tolerance. This high performance is further exhibited on varying initial flight conditions and tracking task types, and on biased sensor noise and atmospheric disturbances, none of which considerably degrade the controller response. This is achieved not having experienced any of these unexpected changes in dynamics, tracking task or flight conditions during training, indicating a high robustness not achievable with model-based controllers. It is believed that SAC's stochastic policy and deep neural networks allowing for better exploration and a higher generalization power, respectively, are the main contributors to this ability.

It is seen, however, to come at the expense of a stable and consistent offline training performance, which makes the development of SAC controllers more difficult but does not jeopardize online performance. This low training reliability is mainly attributed to the various random processes of the SAC algorithm, from network initialization to its stochastic policy. Interestingly, when training on the failed system, i.e. switching from robust to adaptive control, the performance of the SAC controller worsens or remains the same on five out of six failure cases. It is explained by the increased difficulty of training on the failed plant dynamics, given the already low training reliability.

This research contributes to creating a coupled-dynamics model-free flight controller that allows for fault tolerance to various types of unforeseen failures. It is demonstrated that robust control through DRL can, unlike model-based controllers, adapt to various normal and failed flight conditions. While DRL is already widely used for small-scale UAV flight controllers, this research also paves the way for more applications to civil aircraft inner and outer-loop flight controllers. It is expected that DRL methods will be used more broadly in flight control applications to increase fault tolerance and help achieve safe fully-autonomous flight. It is suggested for further research to explore deterministic policy-based Twin-Delayed Deep Deterministic Policy Gradient or on-policy Proximal Policy Optimization DRL algorithms that, at the expense of enhanced exploration, can increase training reliability and may enable safe online learning, thereby removing the need for a plant model during offline learning.

References

[1] International Air Transport Association (IATA), "Loss of Control In-Flight Accident Analysis Report," , 2015.

[2] Stevens, B. L., Lewis, F. L., and Johnson, E. N., *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*, John Wiley & Sons, 2015.

[3] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, MIT Press, 2011.

[4] Bellman, R., "Dynamic Programming," *Science*, Vol. 153, No. 3731, 1966, pp. 34–37.

[5] Ferrari, S., and Stengel, R. F., "Online Adaptive Critic Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 777–786.

[6] Enns, R., and Si, J., "Helicopter Trimming and Tracking Control using Direct Neural Dynamic Programming," *IEEE Transactions on Neural networks*, Vol. 14, No. 4, 2003, pp. 929–939.

[7] Zhou, Y., van Kampen, E.-J., and Chu, Q., "Nonlinear Adaptive Flight Control using Incremental Approximate Dynamic Programming and Output Feedback," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2016, pp. 493–496.

[8] Zhou, Y., van Kampen, E.-J., and Chu, Q. P., "Incremental Model Based Online Dual Heuristic Programming for Nonlinear Adaptive Control," *Control Engineering Practice*, Vol. 73, 2018, pp. 13–25.

[9] Konatala, R., van Kampen, E.-J., and Looye, G., "Reinforcement Learning Based Online Adaptive Flight Control for the Cessna Citation II (PH-LAB) Aircraft," *AIAA Scitech 2021 Forum*, 2021, p. 0883.

[10] Heyer, S., Kroezen, D., and van Kampen, E.-J., "Online Adaptive Incremental Reinforcement Learning Flight Control for a CS-25 Class Aircraft," *AIAA Scitech 2020 Forum*, 2020, p. 1844.

[11] Kroezen, D., "Online Reinforcement Learning for Flight Control," MSc thesis, Delft University of Technology, 2019.

[12] Lee, J., and van Kampen, E.-J., "Online Reinforcement Learning for Fixed-Wing Aircraft Longitudinal Control," *AIAA Scitech 2021 Forum*, 2021, p. 0392.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., "Playing Atari with Deep Reinforcement Learning," *Neural Information Processing Systems Deep Learning Workshop*, 2013.

[14] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous Control with Deep Reinforcement Learning," *International Conference on Learning Representations*, 2016, p. 40.

[15] Tsourdos, A., Permana, I. A. D., Budiarti, D. H., Shin, H.-S., and Lee, C.-H., "Developing Flight Control Policy Using Deep Deterministic Policy Gradient," *2019 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, IEEE, 2019, pp. 1–7.

[16] Sohege, Y., Quinones-Grueiro, M., and Provan, G., "Unknown Fault Tolerant Control using Deep Reinforcement Learning: A Blended Control Approach," *Principles of Diagnostics (DX-19) Conference*, 2019, p. 23.

[17] Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M., "Control of a Quadrotor with Reinforcement Learning," *IEEE Robotics and Automation Letters*, Vol. 2, No. 4, 2017, pp. 2096–2103.

[18] Koch, W., Mancuso, R., West, R., and Bestavros, A., "Reinforcement Learning for UAV Attitude Control," *ACM Transactions on Cyber-Physical Systems*, Vol. 3, No. 2, 2019, pp. 1–21.

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

[20] Lopes, G. C., Ferreira, M., da Silva Simões, A., and Colombini, E. L., "Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning," *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, IEEE, 2018, pp. 503–508.

[21] Bøhn, E., Coates, E. M., Moe, S., and Johansen, T. A., "Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs using Proximal Policy Optimization," *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2019, pp. 523–533.

[22] Fujimoto, S., van Hoof, H., and Meger, D., "Addressing Function Approximation Error in Actor-Critic Methods," *International Conference on Machine Learning*, Vol. 80, 2018, pp. 1587–1596.

[23] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *International Conference on Machine Learning*, 2018, pp. 1856–1865.

[24] Achiam, J., "Spinning Up in Deep Reinforcement Learning, Benchmarks," , 2018. [Online; accessed 15-October-2020].

[25] Barros, G. M., and Colombini, E. L., "Using Soft Actor-Critic for Low-Level UAV Control," *arXiv preprint arXiv:2010.02293*, 2020.

[26] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al., "Soft Actor-Critic Algorithms and Applications," *arXiv preprint arXiv:1812.05905*, 2018.

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

[28] Delahaye, D., Puechmorel, S., Tsiotras, P., and Féron, E., "Mathematical Models for Aircraft Trajectory Design: A Survey," *Air Traffic Management and Systems*, Springer, 2014, pp. 205–247.

[29] Glorot, X., and Bengio, Y., "Understanding the Difficulty of Training Deep Feedforward Neural Networks," *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[30] Ruder, S., "An Overview of Gradient Descent Optimization Algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[31] Ba, J. L., Kiros, J. R., and Hinton, G. E., "Layer Normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[32] Lynch, F. T., and Khodadoust, A., "Effects of Ice Accretions on Aircraft Aerodynamics," *Progress in Aerospace Sciences*, Vol. 37, No. 8, 2001, pp. 669–767.

[33] Grondman, F., Looye, G., Kuchar, R. O., Chu, Q. P., and van Kampen, E.-J., "Design and Flight Testing of Incremental Nonlinear Dynamic Inversion-Based Control Laws for a Passenger Aircraft," *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 0385.

[34] Moorhouse, D. J., and Woodcock, R. J., "Background Information and User Guide for MIL-F-8785C, Military Specification-Flying Qualities of Piloted Airplanes," Tech. rep., Air Force Wright Aeronautical Labs Wright-Patterson AFB OH, 1982.