Hybrid Soft Actor-Critic and Incremental Dual Heuristic Programming Reinforcement Learning for Fault-Tolerant Flight Control

Teirlinck, C.; van Kampen, E.

DOI
10.2514/6.2024-2406

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the AIAA SCITECH 2024 Forum

AIAA SciTech Forum
8-12 January 2024, Orlando, FL
AIAA SCITECH 2024 Forum

10.2514/6.2024-2406

Hybrid Soft Actor-Critic and Incremental Dual Heuristic Programming Reinforcement Learning for Fault-Tolerant Flight Control

C. Teirlinck* Erik-Jan van Kampen†
Delft University of Technology, P.O. Box 5058, 2600GB Delft, The Netherlands

Recent advancements in fault-tolerant flight control have involved model-free offline and online Reinforcement Learning (RL) algorithms in order to provide robust and adaptive control to autonomous systems. Inspired by recent work on Incremental Dual Heuristic Programming (IDHP) and Soft Actor-Critic (SAC), this research proposes a hybrid SAC-IDHP framework aiming to combine adaptive online learning from IDHP with the high complexity generalization power of SAC in controlling a fully coupled system. The hybrid framework is implemented into the inner loop of a cascaded altitude controller for a high-fidelity, six-degree-of-freedom model of the Cessna Citation II PH-LAB research aircraft. Compared to SAC-only, the SAC-IDHP hybrid demonstrates an improvement in tracking performance of 0.74%, 5.46% and 0.82% in nMAE for nominal case, longitudinal and lateral failure cases respectively. Random online policy initialization is eliminated due to identity initialization of the hybrid policy, resulting in an argument for increased safety. Additionally, robustness to biased sensor noise, initial flight condition and random critic initialization is demonstrated.

Nomenclature

$\mathbf{x}, \mathbf{s}, \mathbf{a}$	=	environment state, reinforcement-learning state and environment action vectors
$\mathbf{x}^e, \mathbf{x}^c$	=	environment state error and cost vectors
n, k, m	=	number of reinforcement-learning states, environment states and environment actions
r, γ	=	instantaneous reward and discount factor
τ	=	target smoothing factor
δ	=	temporal difference error
$t, \Delta t, N, T$	=	current time-step, sample time [s], number of samples and simulation time [s]
$f(\mathbf{s}, \mathbf{a})$	=	state transition function
λ_T, λ_S	=	temporal and spacial scaling coefficients
$\pi, \pi_\theta, \mu_\theta, \sigma_\theta$	=	policy, parametric policy and stochastic policy parameterized mean and standard deviation
Q_π, Q_w, V_π	=	action-state value function, parameterized action-state value function and state value function
λ_w	=	parameterized state-derivative of the state value function
θ, w, w'	=	policy, critic and target critic parameter vectors
$\mathcal{H}, \bar{\mathcal{H}}, \eta$	=	entropy, entropy target and temperature coefficient
η_a, η_c	=	actor and critic learning rates
$\mathcal{D}, \mathcal{B}$	=	replay buffer and mini-batch
$L_\pi, L_Q, L_\lambda, L_\eta$	=	loss functions for policy, SAC-critic, IDHP-critic and temperature coefficient
F, G, Θ, Λ, X	=	state, input, parameter, covariance and measurement matrices of the incremental model
$\kappa, \boldsymbol{\epsilon}$	=	incremental model forgetting factor and innovation or error vector
$\delta_e, \delta_a, \delta_r$	=	elevator, aileron and rudder deflection [deg]
p, q, r	=	roll rate, pitch rate and yaw rate [deg/s]
$\alpha, \beta, \theta, \phi, \psi$	=	angle of attack, sideslip angle, pitch angle and heading angle [deg]
V, h	=	airspeed [m/s], altitude [m]

*M.Sc. Student, Control & Simulation, Delft University of Technology.
†Assistant Professor, Control & Simulation, Delft University of Technology.
Code available at `https://github.com/CasperTeirlinck/RLFC-SACIDHP`

I. Introduction

New automation techniques play a vital role in both the safety and economics of current and future aerospace industry needs. Developments in urban air mobility initiatives focusing on (e)VTOL aircraft have a need for novel automation techniques. Many safety and autonomy challenges have to be overcome to make this viable [1]. In commercial air transportation, there is also significant interest in better fault tolerance and automation techniques. Accident rates have dropped significantly over the last two decades, however of all fatal accidents in commercial flights from 2009 to 2018, 60.4% are still caused by in-flight loss of control [2]. Flight control systems that are currently in use mainly use classical control theory. These techniques use linear controllers and require gain scheduling in order to cover the flight envelope of non-linear systems. This gain-scheduling process can be tedious, especially for complex coupled-dynamics systems, and also relies on an accurate plant dynamics model [3]. These classical controllers also lack adaptive behaviour and are not sufficient for increasingly autonomous systems that need to be able to deal with unexpected failures. Hence, there is a need for methods that better handle non-linear systems and can enable fault-tolerant control. Reinforcement Learning (RL) from the field of Machine Learning (ML) [4] is now being researched for adaptive control applications. Traditional tabular RL methods use discrete state and action spaces, which are infeasible for controlling most complex airborne systems. Thanks to the developments in continuous function approximators such as Artificial Neural Networks (ANNs), continuous state and action spaces are possible and several methods have been developed. Adaptive and robust control can be achieved with RL by using online and offline learning techniques. While training online can be highly adaptive, it can also be a safety concern due to a continually changing policy. Instead, RL controllers can also be trained offline, where the generalization power of the function approximators provides robust control.

The field of Approximate Dynamic Programming (ADP) [5] [6] uses mostly shallow ANNs as function approximators and contains the class of Adaptive Critic Designs (ACDs) [7]. These actor-critic designs have been successfully applied in simulation to flight control of a business jet aircraft [8], helicopter tracking control [9] and flight control of a fighter aircraft model[10]. The ADP methods do however still require an accurate model of the plant dynamics in an offline training phase, which limits the ability to deal with random failures. Newer ACDs implement an incremental approach to system identification, which makes them highly adaptive and not reliant on accurate system models. Incremental Heuristic Dynamic Programming (IHDP) [11] and Incremental Dual Heuristic Programming (IDHP) [12] both are incremental methods that can be applied fully online and provide adaptive fault-tolerant control by identifying an incremental plant model in real-time. Recent works [13] [14] [15] on these Incremental Approximate Dynamic Programming (iADP) methods have further explored their applicability to flight control of the PH-LAB research aircraft, but still require more validation and high fidelity simulations before real-world flight tests are feasible. The main advantage for (i)ADP methods is the high sample efficiency, with the potential to react to severe failure cases.

Thanks to the increasing popularity of Deep Learning (DL) in recent years, the advancements made in training Deep Neural Networks (DNNs) have found their way into RL characterizing the field of Deep Reinforcement Learning (DRL). A major development by DeepMind in 2015 used DRL to achieve human-level performance on a number of classic Atari games using a deep Q-network [16]. With a similar achievement in 2017, DeepMind's AlphaGo [17] also demonstrated the ability of DRL algorithms to outperform a human by defeating a world-class GO player. For the continuous control of an aircraft, DRL methods that specifically can handle continuous spate and action spaces can be used. Trust Region Policy Optimization (TRPO) [18] and Deep Deterministic Policy Gradient (DDPG) [19] are two notable extensions of the classic Q-learning approach that provide continuous control. TRPO has been improved on by the state-of-the-art algorithm, Proximal Policy Optimization (PPO) which has been successfully demonstrated in control of a fixed-wing UAV [20] and path generation for an aircraft guidance task[21]. DDPG has also been improved with state-of-the-art algorithms such as Twin-Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC) which improve the learning instability of DDPG. UAV path planning and tracking control have been demonstrated using TD3 [22] [23] and SAC [24] [25] [26]. SAC has recently also been researched in the context of a coupled flight controller for the PH-LAB research aircraft [27], making the SAC controller the baseline for this research. The main advantage of DRL methods is the generalization power of the DNNs and the scalability to high-dimensional spaces.

The contribution of this paper is to advance the development of model-independent, adaptive and robust flight controllers. More specifically, by developing an RL-based flight controller for the Cessna Citation II. This is achieved by presenting a hybrid framework that aims to combine the advantages of the state-of-the-art SAC and IDHP frameworks in providing fault-tolerance to unexpected failures and providing robust flight control.

The theoretical foundations for the SAC and IDHP algorithms are presented in section II followed by the flight controller design in section III. The results of the hybrid method compared to SAC-only are discussed in section IV followed by the conclusion in section V.

II. Fundamentals

This section first formulates the flight control task as a reinforcement learning problem. Additionally, a detailed overview of the two baseline algorithms used in finding a suitable control policy is provided.

A. Reinforcement Learning Problem Formulation

RL frameworks involve working inside the agent-environment interface, which is mathematically described by a Markov Decision Process (MDP). This consists of a RL agent that at time t selects an action $\mathbf{a}_t \in \mathbb{R}^m$ which acts on the environment with state $\mathbf{s}_t \in \mathbb{R}^n$. The next state is determined by the state-transition function as seen in Equation 1 which is governed by the environment dynamics. The MDP has the Markov Property, meaning that the current state and action carry all the information to predict the next state. A scalar reward $r_{t+1} \in \mathbb{R}$ is then determined based on the environment state and used as feedback for the agent. The goal of the agent is to maximize the reward over the time of the episode of N time-steps, or the discounted return G_t defined in Equation 2. The discount factor γ is used to trade-off future to immediate rewards.

Actor-critic RL frameworks consists of an actor that learns the policy, and a critic that learns a value function. The policy can be stochastic as in Equation 3, or deterministic as in Equation 5. Both policy types are used in the hybrid framework presented in this research. Additionally, the critic can estimate an action-value function, or Q-function Q_π as seen in Equation 4. The Q-function maps a given state and action to a scalar value representing the value of being in that state and taking the given action, while following the policy π thereafter. This type of value function is used in the SAC framework. A state value function as seen in Equation 6 works similarly, but only maps a given state to the value of being in that state. The state derivative of the state value function is used in the IDHP framework.

Note that in practice, the state $\mathbf{s}_t$ is often a subset of the full environment state $\mathbf{x}_t$ and/or is augmented with additional samples as shown in section III, however the terms for RL-state and environment state are often used interchangeably.

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

$$G_t = \sum_{k=0}^{N} \gamma^k r_{t+k+1} \tag{2}$$

$$\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t) \tag{3}$$

$$Q_\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\pi}[G_t \mid \mathbf{s}_t, \mathbf{a}_t] \tag{4}$$

$$\mathbf{a}_t = \pi(\mathbf{s}_t) \tag{5}$$

$$V_\pi(\mathbf{s}_t) = \mathbb{E}_{\pi}[G_t \mid \mathbf{s}_t] \tag{6}$$

B. Soft Actor-Critic Framework

Soft Actor-Critic (SAC) is a state-of-the-art offline-learning off-policy DRL algorithm [28]. The main characteristics of SAC are the use of soft policy iteration which includes an entropy term in the policy objective function, and the use of a stochastic policy during training. Hence, a high level of exploration is achieved and the SAC agent is trained offline. When evaluated, the policy is sampled using only the mean of the policy distribution, making it deterministic at evaluation. Since SAC is off-policy, experience replay is used by storing samples in the replay buffer $\mathcal{D}$. Every learning step, a mini-batch $\mathcal{B}$ of experience samples can be sampled from the replay buffer.

1. Actor

The actor learns the SAC policy π_θ, parameterized by the parameter vector θ representing the parameters of a DNN. The stochastic policy distribution is implemented by having two outputs of the policy, being the standard deviation σ_θ and mean μ_θ. This is then used to sample an action from a normal distribution with σ_θ and μ_θ. Note that this sampling requires a “reparameterization trick” to ensure differentiability of the sampled action, which is necessary for the gradient calculations. This is usually implemented using an input Gaussian noise vector ϵ_t and sampling an action using $\mathbf{a}_t = f_\theta(\epsilon_t, \mathbf{s}_t) = \mu_\theta(\mathbf{s}_t) + \epsilon_t \cdot \sigma_\theta(\mathbf{s}_t)$. In [29] this reparameterization is implemented manually, but the implementation of the normal distribution used here applies this under the hood.

The loss function for the policy can be seen in Equation 7. It depends on the Q-function critics and also includes the entropy term with the coefficient η. The log probabilities $\log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$ are also derived from the normal distribution and used in the update rules of both the actor and the critic.

$$L_\pi = \mathop{\mathbb{E}}_{\substack{\mathbf{s}_t\sim\mathcal{B}\\ \mathbf{a}_t\sim\pi}}\left[\min_{i=1,2} Q'_{w'_i}(\mathbf{s}_t,\mathbf{a}_t) - \eta\log\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)\right] \tag{7}$$

In complex control tasks like 6-degree-of-freedom flight control, the action of the SAC policy tends to be highly noisy and oscillatory. A method to smooth out the control input can be implemented by letting the policy control the action increment $\Delta\mathbf{a}$ instead of the action directly [27]. The development of a hybrid framework in this research however requires both the SAC and the IDHP frameworks to operate inside the same control loop. Initial tests showed the IDHP framework has difficulty controlling an action derivative as opposed to the direct action of a complex system. Hence, a different method is chosen to smooth out the policy that is compatible with direct action control using an additional policy regularization term.

Using temporal and spatial regularization terms, described by Conditioning for Action Policy Smoothness (CAPS) [30] forces the policy to keep new actions close to the previous action, and keeps actions close to actions corresponding to similar states. The temporal regularization loss is defined in Equation 8 and computes the distance between the previous and current actions. The spacial regularization loss is defined in Equation 9 and computes the distance between the action and the action based on a normally sampled state $\bar{\mathbf{s}} \sim N(\mathbf{s}, \sigma = 0.05)$. The distances are implemented as the L2-norm. The total CAPS loss term is defined in Equation 10 and includes two additional scaling parameters λ_T and λ_S for the temporal and spacial terms respectively. Note that only the mean, or the deterministic action of the policy is used in computing the distances and not the entire policy distribution

$$L_T = D(\pi(\mathbf{s}_t), \pi(\mathbf{s}_{t+1})) = ||\pi(\mathbf{s}_t) - \pi(\mathbf{s}_{t+1})||_2 \tag{8}$$

$$L_S = D(\pi(\mathbf{s}), \pi(\bar{\mathbf{s}})) = ||\pi(\mathbf{s}) - \pi(\bar{\mathbf{s}})||_2 \tag{9}$$

$$L_\pi^{CAPS} = L_\pi + \lambda_T L_T + \lambda_S L_S \tag{10}$$

2. Critic

The critic learns the Q-function and in this case consists of two separate critics Q_{w_i} with their respective target critics $Q'_{w'_i}$ and parameters vectors w_i and w'_i for $i \in [1, 2]$. Each Q-function is updated separately using its own loss function, and the minimum of the two Q-values is used in the update rules to prevent overestimation. The target critics are used to slow down the gradient updates with the purpose of increasing learning stability. This is achieved by updating the target weights according to a soft update mechanism $w'_{t+1} = \tau w_t + (1-\tau)w'_t$ using the smoothing factor τ. The loss function for the critic can be seen in Equation 11. As already seen in the actor update rule, the minimum of the twin target critics is used in the update rule of actor and critic. This is done to prevent over-estimation of the value.

$$L_{Q_i} = \mathop{\mathbb{E}}_{\substack{(\mathbf{s}_{t+1},\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{B}\\ \mathbf{a}_{t+1}\sim\pi}}\left[\left(Q_{w_i}(\mathbf{s}_t,\mathbf{a}_t) - \left(r_{t+1} + \gamma\left(\min_{i=1,2} Q'_{w'_i}(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) - \eta\log\pi_\theta(\mathbf{a}_{t+1} \mid \mathbf{s}_{t+1})\right)\right)\right)^2\right] \tag{11}$$

3. Entropy

The SAC framework tries to maximize entropy in addition to maximizing the expected return. This ensures a high level of exploration. The entropy of the policy distribution is defined in Equation 12.

$$\mathcal{H}(\pi_\theta(\cdot \mid \mathbf{s}_{t+1})) = \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi}\left[-\log\pi_\theta(\mathbf{a} \mid \mathbf{s}_t)\right] \tag{12}$$

In the update rules of the actor and critic, the entropy term is weighted using the entropy coefficient or temperature coefficient η. The SAC algorithm has been shown to be highly sensitive to the temperature coefficient, thus it was proposed by [28] to learn η automatically. The loss function for this automatic learning process can be seen in Equation 13. The term $\bar{\mathcal{H}}$ is a constant entropy target and is set to the negative of the action space size [29]. In the case of an aircraft environment with three control surfaces, $\bar{\mathcal{H}} = -3$.

$$L_\eta = \mathop{\mathbb{E}}_{\substack{\mathbf{s}_t\sim\mathcal{B}\\ \mathbf{a}_t\sim\pi}}\left[-\eta\log\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) - \eta\bar{\mathcal{H}}\right] \tag{13}$$

4. Overview

In Figure 1, an overview of the SAC framework can be seen showing the interactions between the actor, critic, entropy and environment. Data flow is depicted by solid arrows, while update processes are shown using dashed arrows. The off-policy design is made clear by having two kinds of forward passes through the policy, one where the environment observation generates a new action to take every time-step, and one where the replay buffer is sampled to perform the updates. Also, different notations of the replay buffer signals are used where $\{a_t\}$ is an action batch sampled from the replay buffer and $\{a_t\} \sim \pi$ is a batch of newly generated actions using the observations from the replay buffer. The policy and Q-functions are modelled using DNNs. The update rules described above update the network parameters according to Stochastic Gradient Descent (SGD) using the gradients $\nabla_\theta L_\pi$, $\nabla_{w_i} L_{Q_i}$ and $\nabla_\eta L_\eta$ for the actor, critics and temperature coefficient respectively.

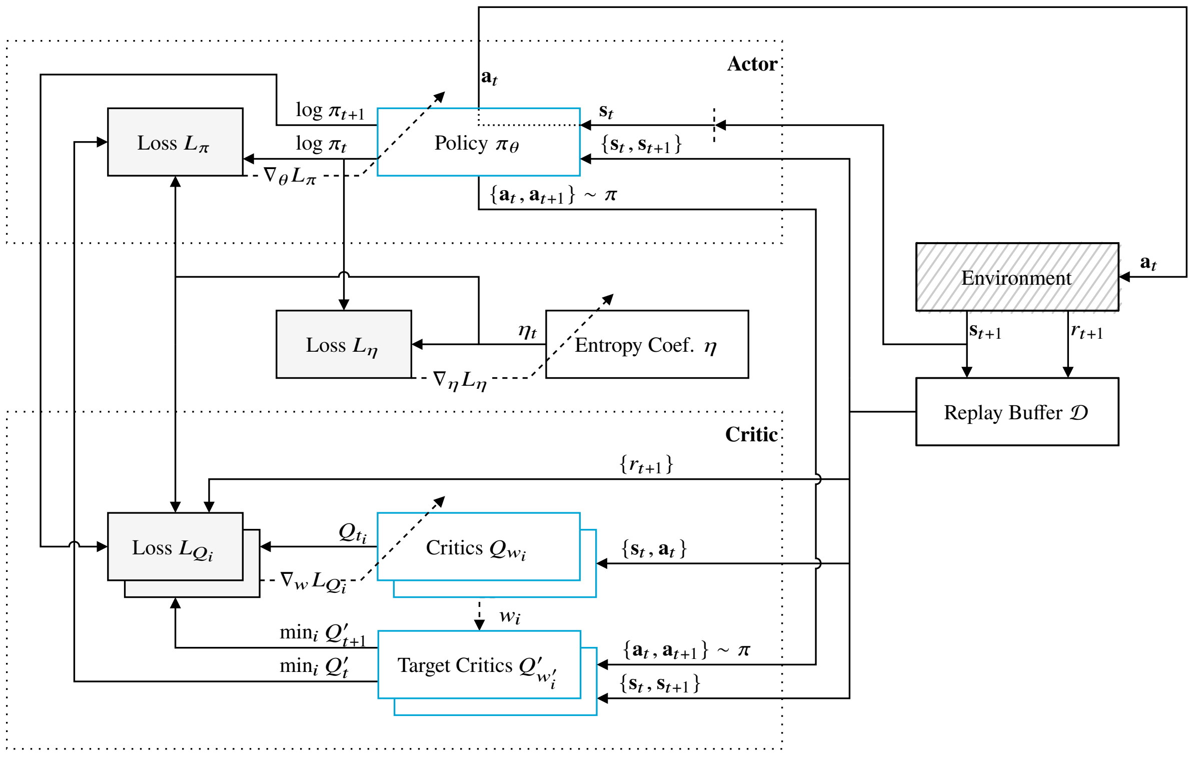


Fig. 1 SAC framework architecture, adapted from [27]

C. Incremental Dual Heuristic Programming Framework

Incremental Dual Heuristic Programming (IDHP) is a state-of-the-art online on-policy ACD algorithm [7]. IDHP is characterized by a linearized, time-varying incremental model of the environment. This model is estimated online and part of the learning process. The agent assumes no prior knowledge of the environment dynamics, hence this method is still considered model-free in the context of this research. Furthermore, the policy is deterministic and the critic consists of the derivative of the state value function as opposed to a Q-function critic in SAC. Compared to SAC, the sample efficiency is considerably higher, assuming a sufficiently high sampling rate [31], and this method can be trained online providing an adaptive fault-tolerant control policy.

Note that the distinction between the RL-state **s** and the environment state **x** is important to make here. Since **s** is often only a subset or augmented version of **x** by design, the state vector used by the partial state derivatives and incremental model has to be explicitly defined as the environment state vector **x** in order for the incremental model to retain a meaningful estimation of the system dynamics.

1. Actor

The actor learns the deterministic IDHP policy π_θ parameterized by the parameter vector θ. The loss function for the policy can be seen in Equation 14 and consists of the next Bellman value estimate with γ the discount factor. Since the output of the critic is the state partial derivative of V, the gradient of L_π does not need backpropagation through the critic network and can use the output of the critic directly in the gradient. The gradient of the loss function can then be derived as seen in Equation 15 where the critic value comes from the target critic $\lambda'_{w'}$. The term $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{a}_t}$ can be replaced by the incremental model input matrix G_{t-1} as per definition of the input matrix according to the model discussed in section II.C.3. The term $\frac{\partial \mathbf{a}_t}{\partial \theta}$ is calculated using backpropagation on the actor.

$$L_\pi = -V(\mathbf{s}_t) = -\left[r_{t+1} + \gamma V(\mathbf{s}_{t+1})\right] \tag{14}$$

$$\begin{aligned}\nabla_\theta L_\pi = \frac{\partial L_\pi}{\partial \theta} &= -\left[\frac{\partial r_{t+1}}{\partial \mathbf{x}_{t+1}} + \gamma \lambda'_{w'}(\mathbf{s}_{t+1})\right] \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{a}_t} \frac{\partial \mathbf{a}_t}{\partial \theta} \\ &= -\left[\frac{\partial r_{t+1}}{\partial \mathbf{x}_{t+1}} + \gamma \lambda'_{w'}(\mathbf{s}_{t+1})\right] G_{t-1} \frac{\partial \mathbf{a}_t}{\partial \theta}\end{aligned} \tag{15}$$

2. Critic

The IDHP critic estimates the partial derivative of the state value function with respect to the state $\lambda_w(\mathbf{s}_t) = \frac{\partial V(\mathbf{s}_t)}{\partial \mathbf{x}_t}$ with parameter vector w.

The loss is defined as the mean squared error of the state derivative of the TD error $\frac{\partial \delta_t}{\partial \mathbf{x}_t}$ as seen in Equation 16. In Equation 17, the formulation of the TD error for the critic can be seen, which corresponds to the next value function estimate called the TD target minus the current value estimate $V(\mathbf{s}_t)$. Taking the state partial derivative of the TD error results in Equation 18 where the TD target is calculated using the target critic value $\lambda'_{w'}$. The state derivative of the reward is provided by the environment, while the term $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}$ can be computed by using the incremental model as seen in Equation 19. The term $\frac{\partial \mathbf{a}_t}{\partial \mathbf{x}_t}$ or $\frac{\partial \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\partial \mathbf{x}_t}$ can be obtained by backpropagation through the policy network. The loss gradient can then be derived using these previous definitions, as seen in Equation 20. The target critics are updated according to the same smooth update method used in SAC.

$$L_\lambda = \frac{1}{2}\left(-\frac{\partial \delta_t}{\partial \mathbf{x}_t}\right)\left(-\frac{\partial \delta_t}{\partial \mathbf{x}_t}\right)^T \tag{16}$$

$$\delta_t = r_{t+1} + \gamma V(\mathbf{s}_{t+1}) - V(\mathbf{s}_t) \tag{17}$$

$$\frac{\partial \delta_t}{\partial \mathbf{x}_t} = \left[\frac{\partial r_{t+1}}{\partial \mathbf{x}_{t+1}} + \gamma \lambda'_{w'}(\mathbf{s}_{t+1})\right] \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t} - \lambda_w(\mathbf{s}_t) \tag{18}$$

$$\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t} = F_{t-1} + G_{t-1}\frac{\partial \mathbf{a}_t}{\partial \mathbf{x}_t} \tag{19}$$

$$\nabla_w L_\lambda = \frac{\partial L_\lambda}{\partial w} = \frac{\partial L_\lambda}{\partial \lambda_w(\mathbf{s}_t)} \frac{\partial \lambda_w(\mathbf{s}_t)}{\partial w} = -\frac{\partial \delta_t}{\partial \mathbf{x}_t} \frac{\partial \lambda_w(\mathbf{s}_t)}{\partial w} \tag{20}$$

3. Incremental Model

The incremental model provides a future estimate of the environment state to be used in the update rules for the actor and critic. This model is derived from a first-order Taylor series expansion [32] and can be seen in Equation 25. Here, the state matrix F_{t-1} and input matrix G_{t-1} are time-varying and are updated every time-step using an RLS estimator.

The RLS update rule of the incremental model can be seen in Equation 22 with Θ the parameter matrix as defined in Equation 21 and $\kappa \in [0, 1]$ the forgetting factor. The measurement matrix X contains the increments of the previous state and action as seen in Equation 24. The error or innovation ϵ is defined in Equation 26 and represents the prediction error between the actual state and the predicted state. Finally, the covariance matrix Λ estimates a measure of the covariance of the parameter estimates and is updated according to Equation 23. Both the parameter matrix and covariance matrix are expressed recursively and thus need an initial value. In this case, the parameter matrix is initialized as zero's and the covariance matrix as an identity matrix of magnitude Λ_0 as no prior knowledge of the parameter covariances is assumed. The magnitude Λ_0 is usually set to a large value, as the uncertainty of the parameters is high at the initial stage. The state and input matrices of the incremental model are used in the update rules of both the actor and the critic.

$$\Theta_{t-1} = \begin{bmatrix} F_{t-1}^T \\ G_{t-1}^T \end{bmatrix} \quad (21) \qquad \Theta_t = \Theta_{t-1} + \frac{\Lambda_{t-1}X_t}{\kappa + X_t^T\Lambda_{t-1}X_t}\boldsymbol{\epsilon}_t \quad (22) \qquad \Lambda_t = \frac{1}{\kappa}\left[\Lambda_{t-1} - \frac{\Lambda_{t-1}X_tX_t^T\Lambda_{t-1}}{\kappa + X_t^T\Lambda_{t-1}X_t}\right] \quad (23)$$

$$X_t = \begin{bmatrix} \Delta\mathbf{x}_t \\ \Delta\mathbf{a}_t \end{bmatrix} \quad (24) \qquad \Delta\mathbf{x}_{t+1} = F_{t-1}\Delta\mathbf{x}_t + G_{t-1}\Delta\mathbf{a}_t \quad (25) \qquad \boldsymbol{\epsilon}_t = \Delta\mathbf{x}_{t+1}^T - \Delta\hat{\mathbf{x}}_{t+1}^T = \Delta\mathbf{x}_{t+1}^T - X_t^T\Theta_{t-1} \quad (26)$$

4. Overview

An overview of the IDHP framework can be seen in Figure 2 which shows the interactions between the actor, critic, incremental model and the environment. Compared to SAC, the actor and critic networks are much shallower and narrower networks, using only a single layer of neurons. The update rules described above update the network parameters according to SGD using the gradients $\nabla_\theta L_\pi$ and $\nabla_w L_\lambda$ for the actor and critic respectively. Additionally, the incremental model is updated using the RLS update rule.

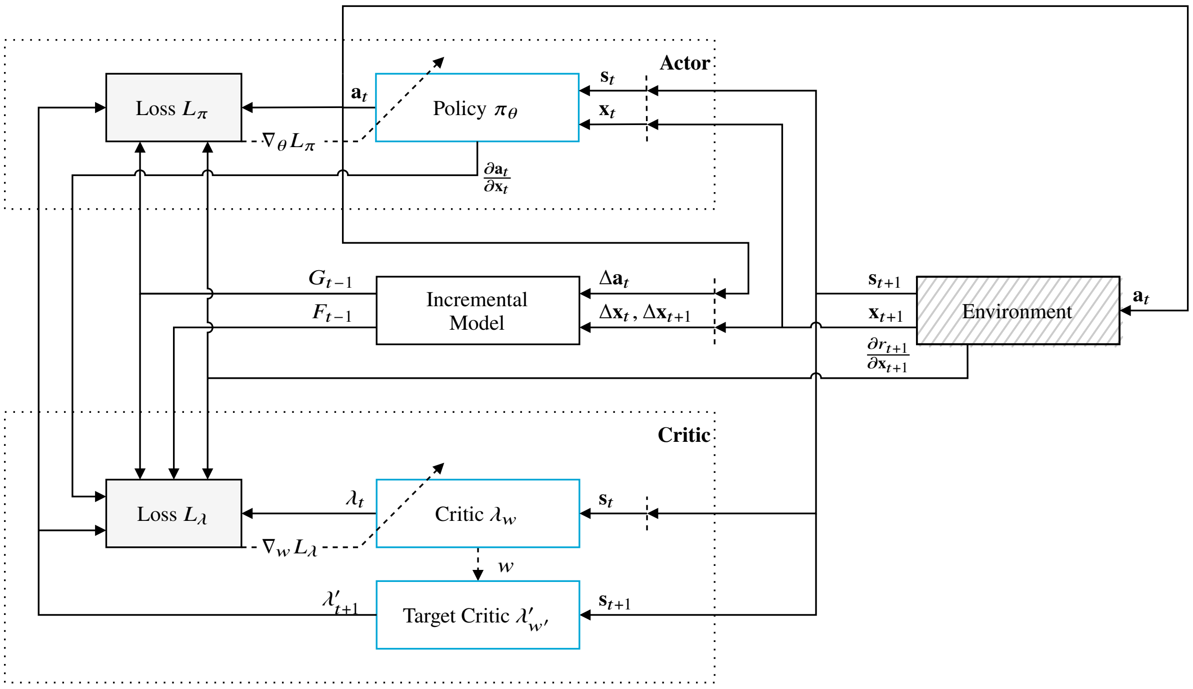


Fig. 2 IDHP framework architecture, adapted from [13][14][15]

III. Flight Controller Design

The flight controller design involves integrating the RL algorithms into a suitable altitude control loop designed to interface with a simulation model of the PH-LAB research aircraft.

A. High-Fidelity Environment Model

The environment is modelled by a high-fidelity non-linear fully-coupled simulation model of the Cessna Citation 500 jet aircraft, built using the DASMAT tool by the Delft University of Technology [33] based on real world flight data. This model can be considered equivalent to the Cessna 550 Citation II PH-LAB aircraft, which is the target platform for the developed controller, despite the difference in fuselage size, engine power and wing size [34]. All simulations are performed with the controller and environment model running at $100Hz$.

The environment state $\mathbf{x} \in \mathbb{R}^k$ and input vector $\mathbf{a} \in \mathbb{R}^m$ can be seen in Equation 27 and Equation 28 respectively. The environment state $\mathbf{x}$ used throughout this paper is the observed state as seen by the RL agent, while the full aircraft state available from the simulation model is denoted by **x'**. A clean configuration is used for all simulations. Additionally, a yaw damper and auto-throttle are provided by the simulation model. The auto-throttle tries to maintain a constant airspeed set by the initial flight condition. The initial control inputs are always untrimmed and zero at the start of a simulation.

$$\mathbf{x'} = [p, q, r, V, \alpha, \beta, \theta, \phi, \psi, h]^T \Rightarrow \mathbf{x} = [p, q, r, \alpha, \theta, \phi, \beta, h]^T \quad (27) \qquad \mathbf{a} = [\delta_e, \delta_a, \delta_r]^T \quad (28)$$

B. Network Architecture

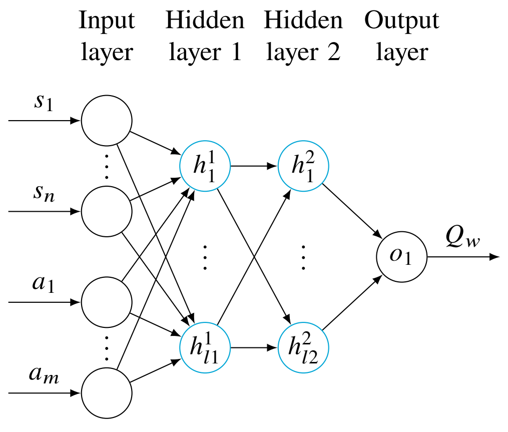
The following two sections describe in more detail the neural network architecture of the SAC and IDHP actors and critics. The SAC network architectures are used in the offline SAC agent of the attitude and altitude controllers, while the SAC-IDHP networks are only used in the inner attitude controller of the final controller structure during online learning.

1. SAC Network Architecture

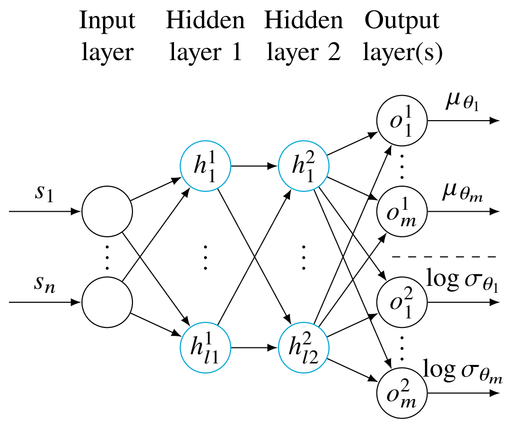
The network topology of actor and critic can be seen in Figure 3. Hidden layer neurons are identified as h and output layers by o with superscript for layer number and subscript for neuron number.

The network of a single Q-function critic in Figure 3a takes both the RL-state and the action as inputs per definition of the Q-function with a single scalar output. The policy network in Figure 3b is constructed using two separate output layers μ_θ and $\log \sigma_\theta$ for the policy mean and standard deviation respectively. The network estimates the log of the standard deviation in order to stay in $\mathbb{R}$ and the exponential is taken in order to build the policy distribution.

Both the actor and critic contain two hidden layers of sizes $l1$ and $l2$. Each hidden neuron h consists of a weighted linear combination of the input vectors with an additional bias term. The signal is subsequently passed to a LayerNormalization layer [35] and finally passed to a ReLU activation function. The output neurons have a linear activation function. All the weights, biases and normalization parameters form the parameter vectors θ and w_i for the actor and critics respectively.



(a) SAC critic Q-function architecture



(b) SAC policy Architecture

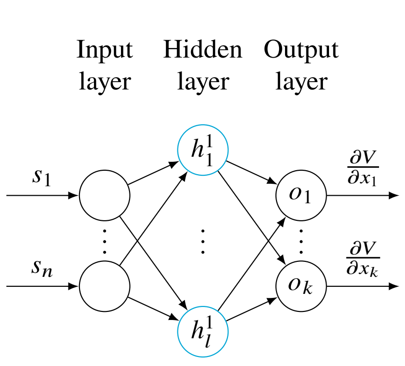
Fig. 3 SAC network architectures

2. Hybrid SAC-IDHP Network Architecture

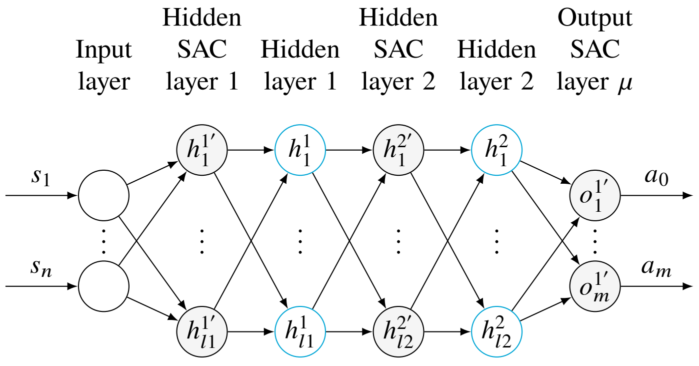
The novelty of the hybrid SAC-IDHP controller developed in this research lies in large part in the network structure of the hybrid policy, as seen in Figure 4b. Contrary to a traditional policy network used in IDHP agents, the hybrid policy includes the pre-trained layers $h^{1'}$, $h^{2'}$ and $o^{1'}$ corresponding to h^1, h^2 and o^1 of the SAC policy. For the output layer, only the policy mean output of the SAC policy is used. The parameters of these pre-trained SAC layers are frozen during the IDHP learning process. Furthermore, the IDHP hidden units h^1 and h^2 do contain learnable parameters consisting of only weights and no bias. Similarly to the SAC policy, the activation function used on the additional hidden layers are ReLU functions, but no LayerNormalization is used in the IDHP neurons. Linear activation functions are used on the output neurons. The hybrid policy parameters vector thus only contains the weights of the hidden units h^1 and h^2.

By constructing the hybrid policy in this way, the goal is to maintain as much of the pre-learned information of the robust SAC policy during online learning. In order to achieve this, the learnable IDHP policy layers are initialized using the identity matrix as opposed to random initialization.

The critic on the other hand relates to a more traditional network structure used in Dual Heuristic Programming as seen in Figure 4a consisting of a single hidden layer. The IDHP critic accepts the RL-state as input and estimates the partial state derivative of the state value function $\frac{\partial V}{\partial \mathbf{x}}$ with k elements. The hidden units are constructed identically to the policy hidden units but with hyperbolic tangent activation functions, while the output neurons also have a linear activation function. The initialization of the critic weights is random as opposed to the actor and sampled from a truncated normal distribution with standard deviation σ_c.



(a) IDHP critic architecture



(b) Hybrid IDHP-SAC combined policy architecture

Fig. 4 Hybrid IDHP-SAC network architectures

C. Attitude Controller

The attitude or inner control loop tracks reference signals for pitch, roll and sideslip angles as seen in Equation 29, and outputs the environment action vector. The tracking error vector can then be defined as in Equation 30 where P is a binary selection matrix defined in Equation 32 mapping the aircraft state to the tracked states. In order to keep the error signals in similar order of magnitude, the scaling vector from Equation 31 is determined by trial and error, where the sideslip signal receives a larger scale due to its lower overall magnitude resulting from the zero-sideslip hold task.

$$\mathbf{x}^{r^{att}} = [\theta^r, \phi^r, \beta^r]^T \quad (29)$$

$$\mathbf{x}_t^{e^{att}} = \mathbf{x}_t^{r^{att}} - P^{att}\mathbf{x}_{t+1} = [\theta^r - \theta, \phi^r - \phi, \beta^r - \beta]^T \quad (30)$$

$$\mathbf{x}^{c^{att}} = \frac{180}{\pi}\left[\frac{1}{30}, \frac{1}{30}, \frac{1}{7.5}\right]^T \quad (31)$$

$$P^{att} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (32)$$

1. SAC Attitude Controller

In the context of SAC, a reward function and RL-state vector are defined by Equation 33 and Equation 34 respectively. The reward function represents the negative of the L1 norm of the scaled error vector clipped on the $[-1, 1]$ interval. The RL-state vector consists of the scaled error vector to ensure good steady-state response and additionally the pitch, roll and yaw rates for improved transient response [27].

$$r_{t+1}^{SAC^{att}} = -\frac{1}{3}\left\|\text{clip}\left[\mathbf{x}_t^{e^{att}} \odot \mathbf{x}^{c^{att}}, \vec{\mathbf{-1}}, \vec{\mathbf{1}}\right]\right\|_1 \quad (33)$$

$$\mathbf{s}_{t+1}^{SAC^{att}} = \left[p, q, r, \left(\mathbf{x}_t^{e^{att}} \odot \mathbf{x}^{c^{att}}\right)^T\right]^T \quad (34)$$

2. IDHP Attitude Controller

The IDHP framework utilizes a reward function defined as the squared scaled error vector, defined in Equation 35. The RL-state for the IDHP critic is defined in Equation 36 and contains the three body rates, pitch, roll, sideslip and angle of attack angles and the scaled error vector. Note that the IDHP framework requires the state derivative of the reward function as defined in Equation 37 which can be derived directly using the definitions of the reward function and error vector.

Note that the RL-state vector for the IDHP actor is the same as the state vector for the SAC actor due to the hybrid policy architecture, and that the environment state vector for the incremental model excludes the altitude from the vector defined in Equation 27.

$$r_{t+1}^{IDHP} = -\left[\mathbf{x}_t^{e^{att}}\right]^T \left[diag\ \mathbf{x}^{c^{att}}\right] \left[\mathbf{x}_t^{e^{att}}\right] \quad (35)$$

$$\mathbf{s}_{t+1}^{IDHP} = \left[p, q, r, \alpha, \theta, \phi, \beta, \left(\mathbf{x}_t^{e^{att}} \odot \mathbf{x}^{c^{att}}\right)^T\right]^T \quad (36)$$

$$\frac{\partial r_{t+1}^{IDHP}}{\partial \mathbf{x}_{t+1}} = 2\left[\mathbf{x}_t^{e^{att}}\right]^T \left[diag\ \mathbf{x}^{c^{att}}\right] P^{att} \quad (37)$$

D. Altitude Controller

The altitude or outer control loop only tracks the altitude reference signal and outputs the reference signal for the pitch angle. The reference vector is defined in Equation 39 with the error vector in Equation 39 using the selection matrix from Equation 41. Also, the altitude error signal requires a scaling factor, as seen in Equation 40.

$$\mathbf{x}^{r^{alt}} = [h^r] \quad (38)$$

$$\mathbf{x}_t^{e^{alt}} = \mathbf{x}_t^{r^{alt}} - P^{alt}\mathbf{x}_{t+1} = [h^r - h,] \quad (39)$$

$$\mathbf{x}^{c^{alt}} = \left[\frac{1}{240}\right] \quad (40)$$

$$P^{alt} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (41)$$

1. SAC Altitude Controller

The outer control loop only implements the SAC algorithm with a reward function similar to the SAC attitude controller as seen in Equation 42. The RL-state vector defined in Equation 43 is simpler than the attitude controller, with only containing the scaled error vector. This is because only the kinematic relationship between the pitch angle and altitude has to be learned.

$$r_{t+1}^{SAC^{alt}} = -\left\|\text{clip}\left[\mathbf{x}_t^{e^{alt}} \odot \mathbf{x}^{c^{alt}}, \vec{\mathbf{-1}}, \vec{\mathbf{1}}\right]\right\|_1 \quad (42)$$

$$\mathbf{s}_{t+1}^{SAC^{alt}} = \left[\mathbf{x}_t^{e^{alt}} \odot \mathbf{x}^{c^{alt}}\right] \quad (43)$$

E. Hybrid SAC-IDHP Cascaded Controller

In Figure 5 a control loop diagram can be seen of the complete cascaded SAC-IDHP hybrid altitude-attitude controller. Only the inner attitude controller implements the hybrid architecture, as it is assumed the majority of dynamic relations that will change during failure modes are learned in the inner loop. Hence, an adaptive element is most useful in the inner loop and the outer loop is only trained offline with the SAC algorithm in order to limit the overall complexity. The dotted lines in the hybrid attitude controller represent the signal flow during online operation, where the IDHP attitude controller controls the aircraft and the SAC attitude controller only provides its pre-learned policy weights.

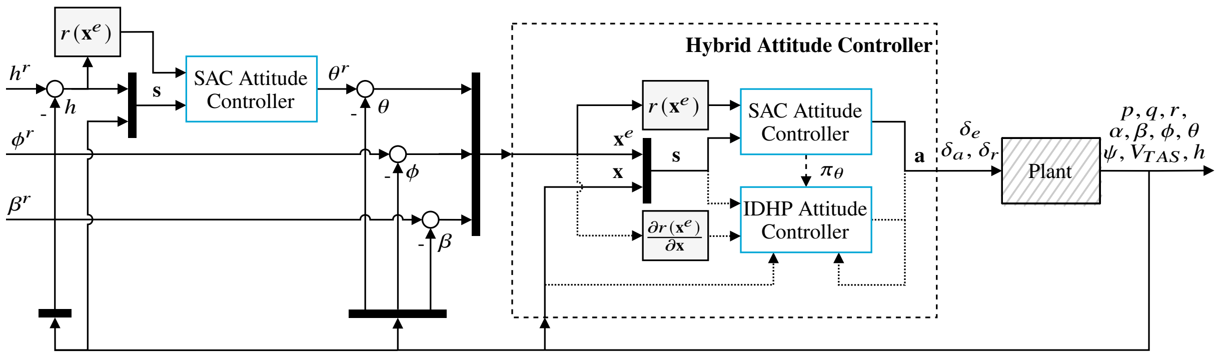


Fig. 5 SAC-IDHP Cascaded Altitude and Attitude Controller Structure

F. Training Strategy

The hybrid RL controller involves multiple stages of training. The SAC attitude and altitude controllers are trained offline, while the SAC-IDHP attitude controller requires an initial online training phase. The training strategy and hyperparameters used for all phases are discussed in this section.

1. Offline SAC Training

The SAC attitude agent is trained first using a maximum of 1000000 time steps. This corresponds to 500 $20s$ training episodes with a double step reference signal for the pitch and roll angles. The magnitude and sign of the step signals is uniformly sampled from $[20°, 10°]$ and $[40°, 20°]$ for the pitch and roll angles, respectively. After every training episode, the agent is evaluated using the same task, but without the randomized reference signal magnitudes and using the maximum values instead. The sideslip reference signal is always held at zero. A batch of at least 5 agents with differing random seeds is trained, where the best performing agent is selected to train the altitude controller.

The altitude controller follows the same training strategy with alternating climb, descend and altitude hold reference signals using $50m$ altitude differences. The roll and sideslip reference signals are equal to the attitude training task.

The hyperparameters used for both SAC controllers can be seen in Table 1 consisting of default values from the original SAC papers and empirically determined values. The parameters $\gamma, \tau, l1, l2, \eta_a, \eta_c, |\mathcal{B}|$ and η_0 are taken from the previously successful SAC implementation [27], while the maximum replay buffer size $|\mathcal{D}|$ has been increased compared to that paper as better learning stability was observed with a larger buffer size. The CAPS regularization coefficients λ_T and λ_S have been determined by trial and error, whereby a trade-off is made between increased smoothness of the policy's action, and decreased tracking performance with increasing coefficient values. The altitude controller requires smaller CAPS scaling coefficients, likely due to the decreased learning complexity of the outer altitude control. Note that the learning rates are linearly decreased to 0 over the total number of time steps.

In Figure 6 the reward curves for attitude and altitude controllers can be seen, showing the average and interquartile ranges over 5 random seeds of converged runs. The attitude and altitude controllers plateau at around -250 and -30 respectively, with little improvements after the initial 200000 time steps.

Compared to the equivalent training curves shown in previous research on the same system [27] the interquartile range of the current results is considerably smaller. This improved learning stability is assumed to be caused by the utilization of direct control and CAPS regularization as opposed to an incremental control approach in the previous research.

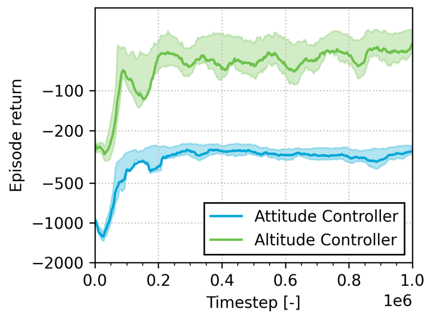


Fig. 6 SAC offline training curves of attitude and altitude controllers. Mean smoothed by a window of size 20 and the interquartile range over 5 random seeds are shown by the solid lines and shaded regions respectively.

Table 1 SAC Hyperparameters, adapted from [27] [28] [29]

Param.	**Value Attitude Agent**	**Value Altitude Agent**	**Description**
γ	0.99	0.99	Discount factor
τ	0.005	0.005	Target critic mixing factor
$l1, l2$	$[64, 64]$	$[32, 32]$	Actor/Critic hidden layer sizes
η_a, η_c	$4.4e-4$	$3.0e-4$	Actor/Critic initial learning rate
$\lvert\mathcal{B}\rvert$	256	256	Replay buffer mini-batch size
$\lvert\mathcal{D}\rvert$	1000000	1000000	Replay buffer maximum size
λ_T, λ_S	400, 400	10, 10	CAPS scaling coefficients
η_0	1.0	1.0	Initial temperature coefficient

2. Online IDHP Training

The IDHP framework requires initial excitation of all the environment states in order to successfully identify the incremental model. In previous IDHP-only frameworks [13] [14], an exponentially decaying sinusoidal excitation signal is added to the agent's action in order to excite the system during the initial training phase. An advantage of the hybrid framework is the presence of the pre-existing converged SAC policy, as obtained in subsubsection III.F.1 at the start of the online learning phase. Because of the identity initialization of the hybrid policy, the initial excitation can be provided by the SAC policy, driven by the reference signals without the use of additional excitation signals. The hybrid attitude controller thus requires an initial online training task using in this case sinusoidal reference signals on the pitch, roll and sideslip angles. The reference signal for the sideslip is decaying in order to better preserve aircraft stability. The IDHP specific hyperparameters used are seen in Table 2, while the SAC portion only involves inference and no updates to the SAC layers are performed during online learning. The IDHP hyperparameters are taken from previous IDHP-only configurations [13] with the exception of the layer size l and learning rates η_a and η_c. The hidden layer size is empirically determined at a small value for increased learning stability, but without observing significant loss in learning ability. The learning rates are determined using trial and error by increasing learning rates until noticeable oscillations or instability appears in the training tasks.

The response on the training task can be seen in Figure 7 where also the SAC-only response is shown. The progression of the actor/critic weights and the incremental model parameters can be seen in Figure 8. Note that weights of identity initialized networks remain near the identity matrix [36], hence the actor weights seen in the parameter plots are plotted separately around 1 and 0. The parameters' progression demonstrates the hybrid method can converge on the training task, with the actor-critic weights and the parameter matrices of the incremental model all converging. The F and G matrices take approximately 8s to converge, while the critic weights are stable after 30s. The actor weights are converging more slowly, but have reached the converging range within the 60s training task.

The nMAE metric as later defined in section IV can already be used here to compare the initial tracking performance with SAC-only. With an nMAE of 7.41% for the hybrid and 10.31% for SAC-only, a noticeable improvement of 2.9% in tracking performance is already demonstrated. Looking at the visual tracking response, after approximately 20s, the hybrid agent has successfully corrected for the steady-state error of the SAC policy with a smaller tracking error near the sinusoidal peaks of the pitch and roll reference signals. The sideslip tracking performance shows little difference, as the SAC policy is already providing a low error. Note that before 20s, the initial converge phase results in temporary divergence from the tracking signal, hence the importance of a controlled training task before performing flight manoeuvre tasks. Also note that the airspeed tracking is handled by the auto-throttle from the DASMAT model and is not managed by the RL-agent.

The resulting weights and parameters are stored and used as initial condition for the hybrid attitude agent in the following flight manoeuvre demonstrations.

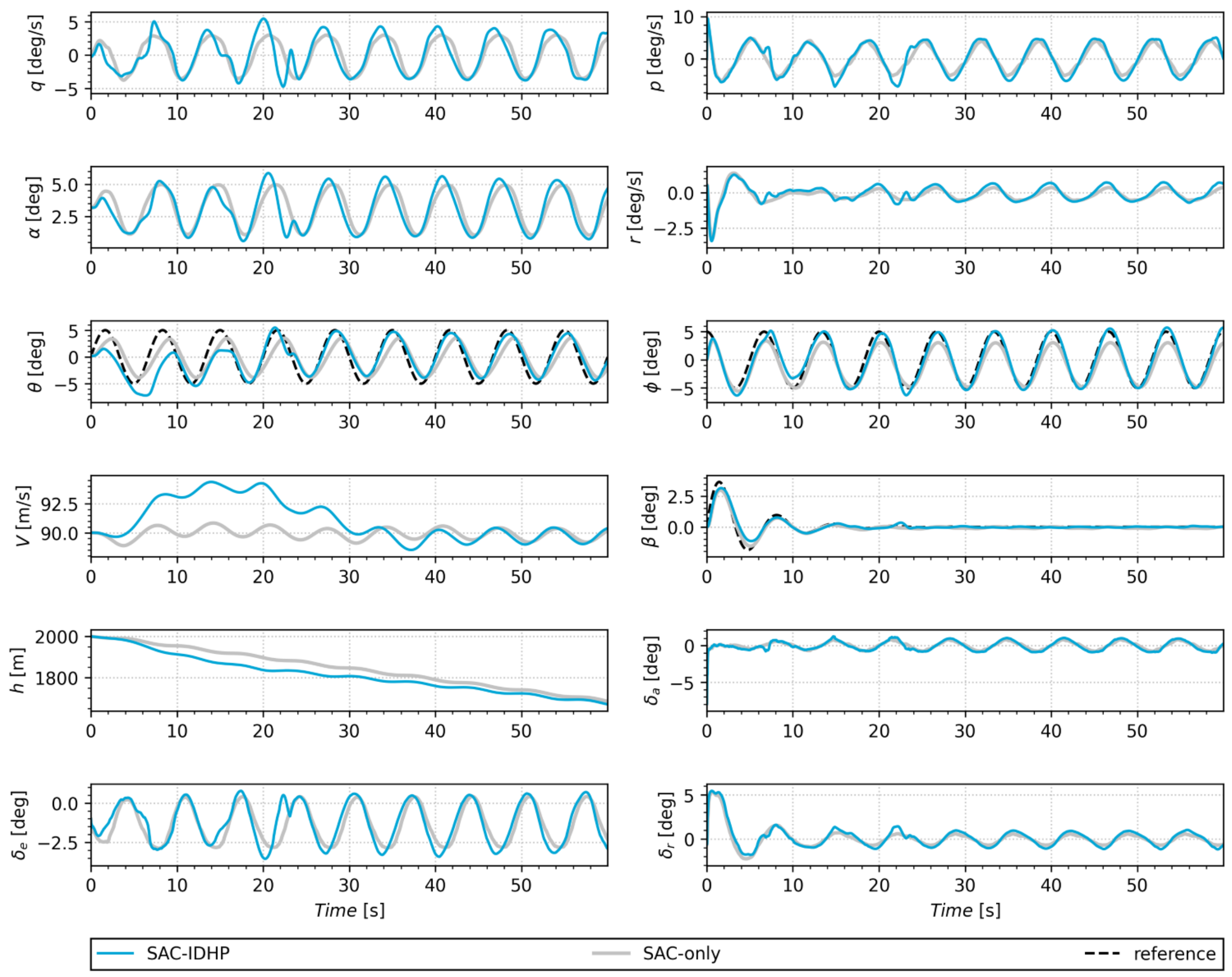


Fig. 7 SAC-IDHP online training response of attitude controller. nMAE = 7.41% for SAC-IDHP and 10.31% for SAC-only.

Table 2 IDHP Hyperparameters, adapted from [13] [14]

Parameter	**Value**	**Description**
γ	0.8	Discount factor
τ	0.01	Target critic mixing factor
l	8	Critic hidden layer size
η_a, η_c	0.2, 1.0	Actor/Critic learning rate
κ	1.0	Incremental model forgetting factor
Λ_0	$1 \cdot 10^8$	Initial covariance matrix magnitude

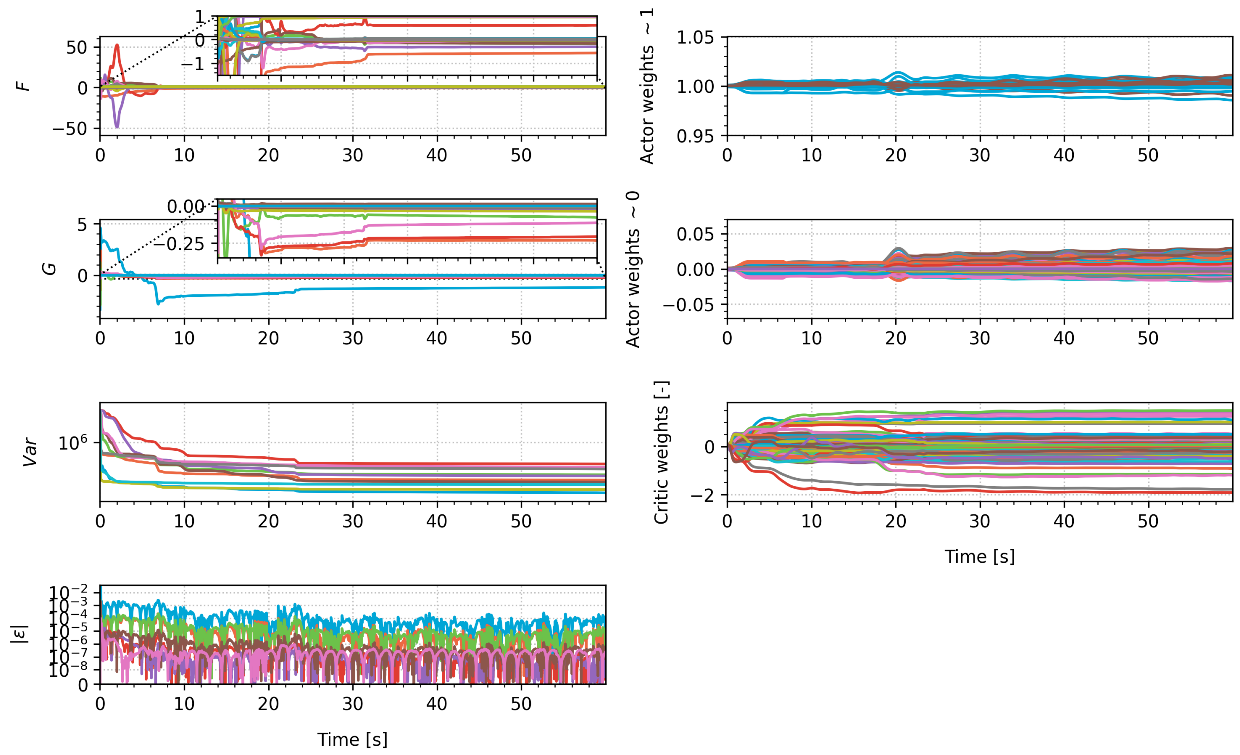


Fig. 8 SAC-IDHP online training of attitude controller, actor/critic weights and incremental model parameters.

IV. Results and Discussion

The response of the Hybrid controller and the SAC-only controller are compared on the simulation model of the Cessna Citation jet aircraft. First, the aircraft in nominal state is evaluated after which several aircraft failures are simulated and the performance between SAC-IDHP and SAC-only is compared. All evaluation runs are performed using an initial condition of $h = 2000m$ and $V = 90\frac{m}{s}$. Additional results concerning varying initial flight conditions, critic initialization and sensor noise are discussed in subsubsection IV.C.1.

A performance metric used in addressing tracking performance is the normalized Mean Absolute Error (nMAE) averaged over externally tracked states which are altitude, roll and sideslip angles for the altitude tasks, and pitch, roll and sideslip angles for the attitude tasks. The normalization is done over the maximum reference signal range with the exception of the sideslip angle, where the normalization range is set at $[-5°, 5°]$ as its reference signal is always 0.

A. Nominal System

The proposed flight controller should be able to control the aircraft in nominal condition without failures. This section presents the control response on the altitude task by comparing the performance of the SAC-only controller against the hybrid SAC-IDHP controller.

In Figure 9 the response on the altitude task can be seen. The external reference signal for the altitude is set at a steady climb and descend over 250m with a 15s hold in between. The bank angle reference is set at alternating 20° and 40° turns motivated by CS-25 specifications for nominal coordinated turns [37]. The sideslip reference is set at zero per definition of a coordinated turn. The pitch angle reference signal shown in the response plots is generated by the SAC outer loop controller based on the altitude error.

Comparing tracking performance, the SAC-only agent achieves a nMAE of 2.77% and the hybrid 2.03% showing a small improvement of 0.74%. Most notably, the sideslip angle has reduced peaks, but the longitudinal states show increased oscillatory behaviour in the hybrid response while keeping closer to the reference signal. The SAC agent shows similar tracking performance to previously developed SAC controllers on the same system [27]. This shows that both SAC-only and the hybrid agent have satisfactory tracking performance on the nominal altitude tracking task.

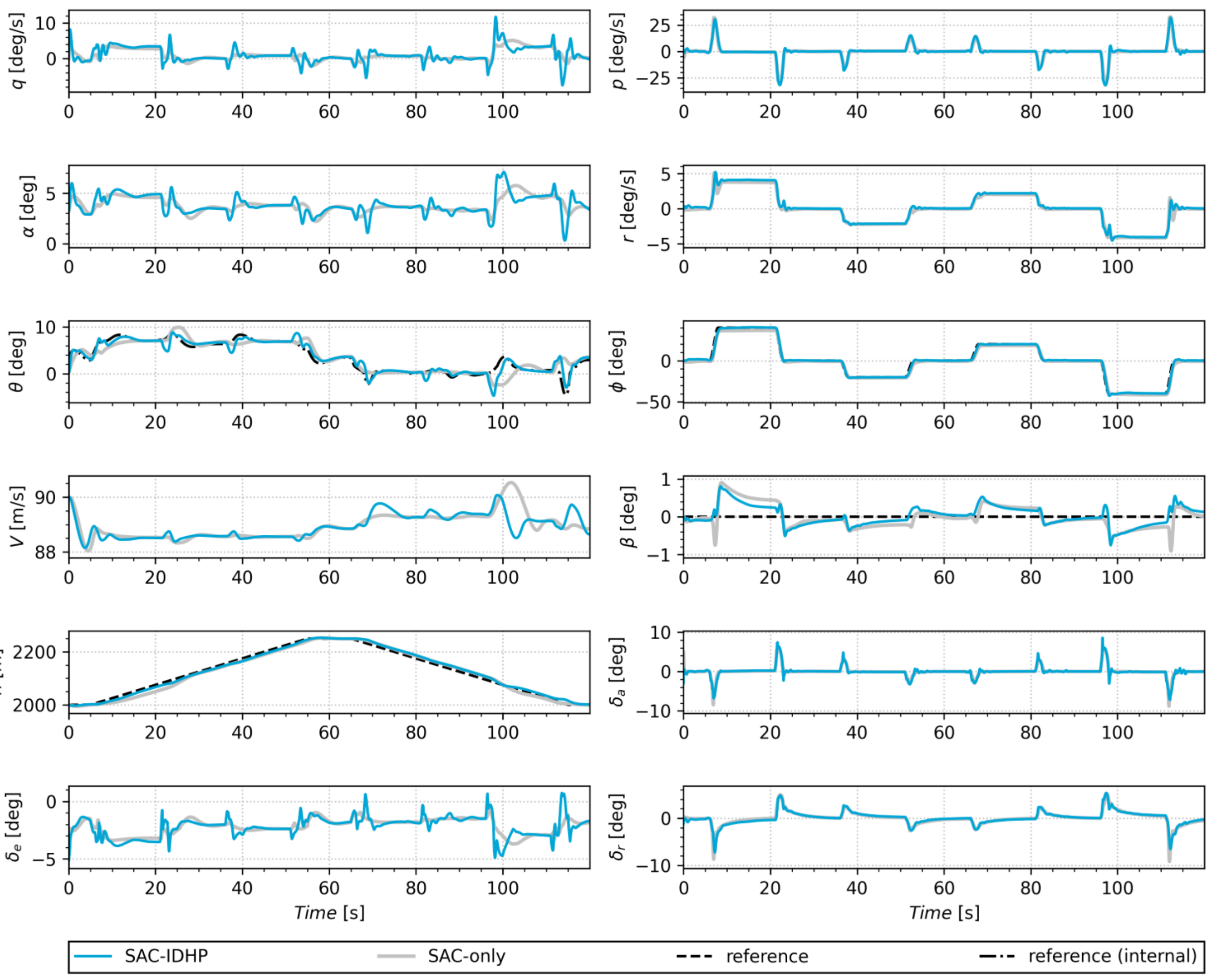


Fig. 9 Altitude tracking response on nominal system. SAC-IDHP and SAC-only compared. nMAE = 2.03% for SAC-IDHP and 2.77% for SAC-only.

B. Failed System

The two most important failure cases tested in this section are the reduced effectiveness of the elevator and ailerons to demonstrate longitudinal and lateral failures respectively. The same reference signals are used as for the nominal case except for the maximum bank angle which is set at 20°.

1. Reduced Elevator Effectiveness

In Figure 10 the response of 70% reduced elevator effectiveness at t=30s can be seen. The SAC agent achieves a nMAE of 7.99% while the hybrid agent maintains an nMAE of 2.53%, an improvement of 5.46%. This shows the hybrid policy is successful in correcting for performance degradation present in the robust response of the SAC policy. Looking at the response, the SAC agent remains stable, but with a considerable tracking error on the altitude due to the heavily reduced elevator effectiveness. The response of the hybrid agent remains close to the altitude reference while also improving on a small steady state error appearing on the roll angle for SAC. Overall, the hybrid agent shows greatly improved tracking performance, but with slightly increased oscillations mainly in the longitudinal states.

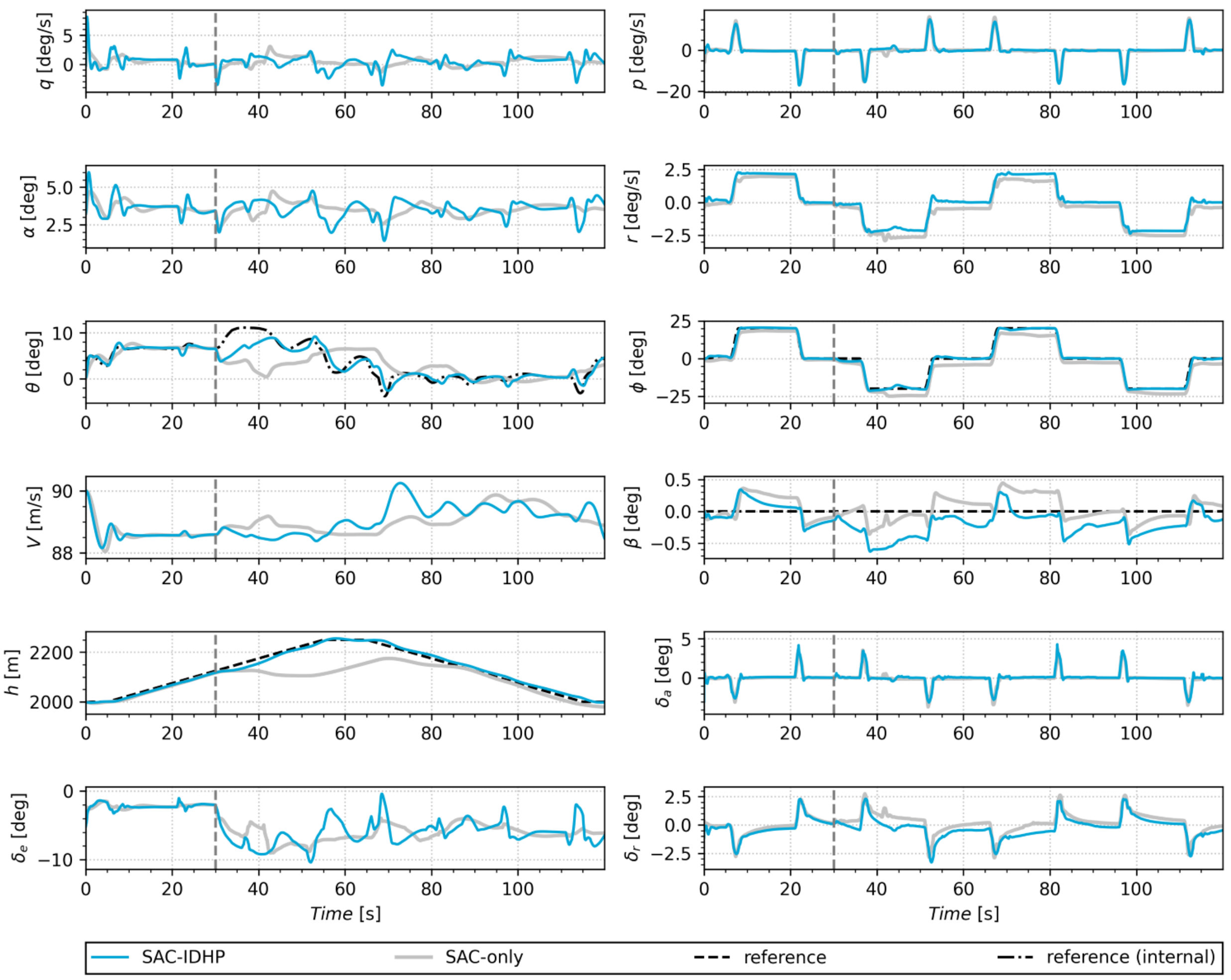


Fig. 10 Altitude tracking response on system with 70% reduced elevator effectiveness from t=30s. SAC-IDHP and SAC-only compared. nMAE = 2.53% for SAC-IDHP and 7.99% for SAC-only.

2. Reduced Aileron Effectiveness

In Figure 11 the response to 90% reduced aileron effectiveness from t=30s can be seen. Comparing tracking error between SAC and SAC-IDHP, the respective nMAE of 3.28% and 2.46% only show a 0.82% improvement for the hybrid agent. This is in line with the nominal case as can also be seen from visual inspection. The robust response of the SAC policy successfully corrects for the reduced elevator effectiveness by increasing the maximum aileron deflection from approximately 4° to 26° and keeping the tracking error small. Nonetheless, the hybrid agent still provides small improvements in rise time of the bank angle, and keeping slightly closer to the zero sideslip reference. Note that for the hybrid agent, increased oscillations are noticeably prevalent in the longitudinal states.

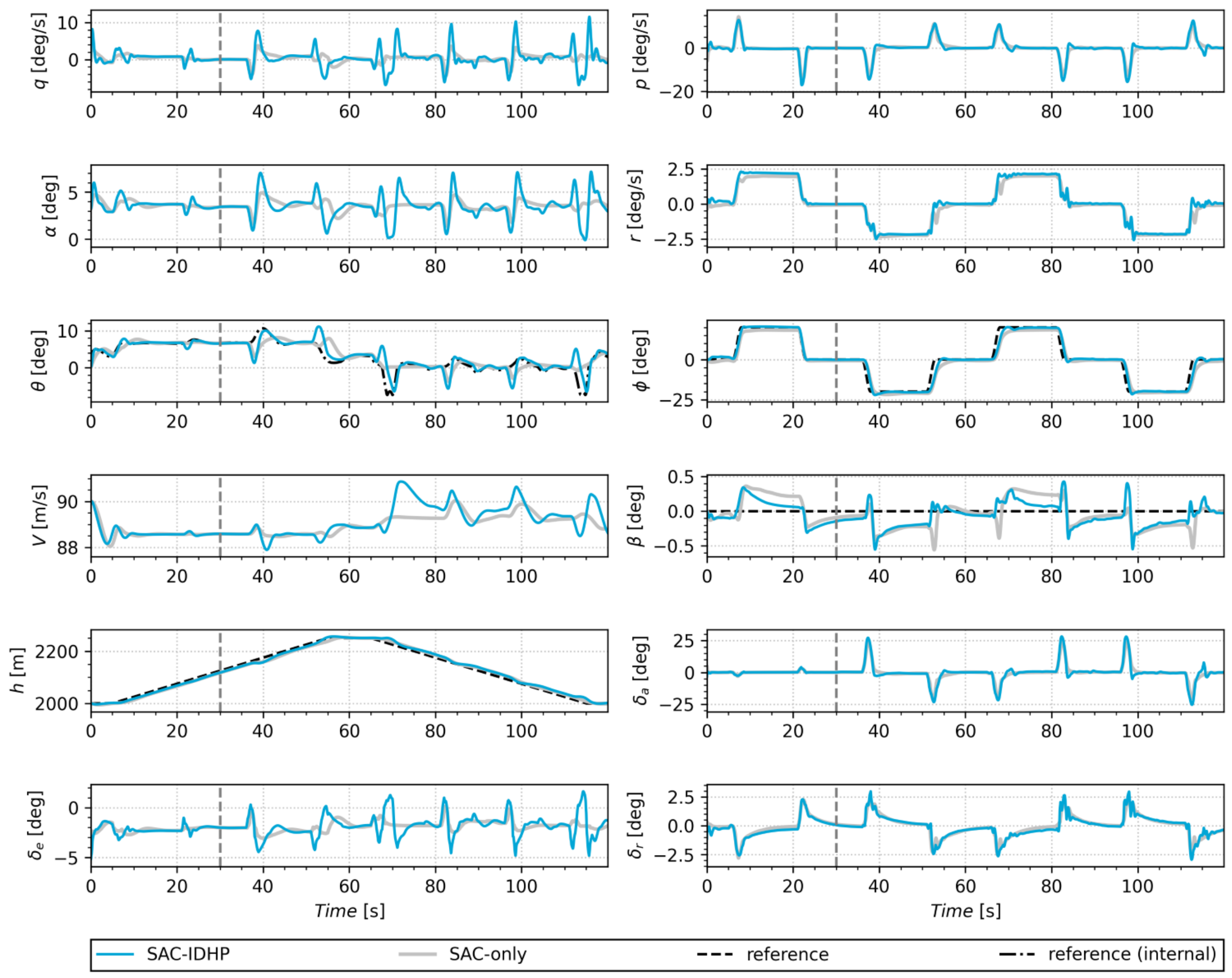


Fig. 11 Altitude tracking response on system with 90% reduced aileron effectiveness from t=30s. SAC-IDHP and SAC-only compared. nMAE = 2.46% for SAC-IDHP and 3.28% for SAC-only.

C. Additional Results

Additional experiments are performed in order to judge the robustness and reliability of the hybrid SAC-IDHP controller compared to a SAC-only controller. Robustness to varying initial flight condition, biased sensor noise and random critic initialization are explored.

1. Robustness to Initial Flight Condition

The SAC offline and the hybrid online training tasks are all performed on an initial condition of $h_0 = 2000m$ and $V_0 = 90\frac{m}{s}$. This section explores variability in tracking performance with changing initial conditions different from the training conditions, all performed on the altitude tracking task with 20° maximum bank angle.

In Table 3 the nMAE for 4 different flight conditions can be seen with FC2 being the nominal condition. The flight conditions are in order of increasing dynamic pressure. It can be seen that tracking error and error variability across the flight conditions is lower for the hybrid method. For FC1 with the lowest dynamic pressure, the SAC agent has the largest error with decreasing error with increasing dynamic pressure, except that FC4 has a slightly higher error than FC2. This pattern is not present for the hybrid method, which has a considerably lower variance and stays within 0.27% nMAE. Note that the scope of this analysis excludes variability over multiple SAC reference policies and IDHP training seeds, the latter being discussed independently in section IV.C.3.

Table 3 Robustness to initial flight conditions of cascaded altitude controllers.

Flight Condition	Initial Altitude [m]	Initial Airspeed [m/s]	nMAE SAC-only	nMAE SAC-IDHP
FC1	5000	90	4.71%	1.96%
FC2 (nominal)	2000	90	2.76%	2.02%
FC3	5000	140	2.05%	1.75%
FC4	2000	140	2.27%	2.01%

2. Biased Sensor Noise

Sensor noise is an essential element in assessing a closer to real-world environment. Biased sensor noise is applied to the observed state using measurement values derived from the PH-LAB aircraft [33] as seen in Table 4. It was noticed during initial testing that the incremental model identification of the IDHP framework produces inconsistent results when high frequency oscillation are present in the states. Hence, a low-pass filter with $\omega_0 = 40deg$ is applied to the observation for the IDHP update rule, with an equivalent filter applied to the SAC-only update rules for a fair comparison. The exact nominal altitude task from subsection IV.A is used, with a resulting nMAE of 2.69% for SAC-only and 2.00% for the hybrid agent. This corresponds to a respective 0.08% and 0.03% reduction in nMAE compared to the case without noise, attributed to the bias having a positive effect on the error, but also indicating both controllers maintain performance in the presence of sensor noise. Note however that the hybrid agent suffers from increased oscillations compared to SAC-only and the case without sensor noise, but shows to have no negative effect on tracking performance. Again, the scope of this analysis excludes variability over IDHP training seeds and SAC reference policies.

Table 4 Cessna Citation PH-LAB sensor noise characteristics [33]

State	Bias μ	Variance σ^2
$p, q, r\ [rad/s]$	$3.0 \cdot 10^{-5}$	$4.0 \cdot 10^{-7}$
$\theta, \phi\ [rad]$	$4.0 \cdot 10^{-3}$	$1.0 \cdot 10^{-9}$
$\beta\ [rad]$	$1.8 \cdot 10^{-3}$	$7.5 \cdot 10^{-8}$
$h\ [m]$	$8.0 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$

3. Sensitivity to Random Critic Initialization

Compared to IDHP-only frameworks, the hybrid framework has a lesser degree online random initialization because of the identity initialization of the IDHP actor layers. The online critic however is still initialized by sampling from a truncated normal distribution with zero mean and a standard deviation σ_c as seen in Equation 44. The effect of this random initialization and the effect of varying standard deviations is evaluated in this section.

All training runs are performed using same hyperparameters from Table 2. A total of 150 runs are performed, 50 for each of three different standard deviations.

An additional metric is used in order to arrive at a total success rate per batch. Relating to the CAPS temporal loss function used in the SAC offline training from Equation 8, a temporal loss metric is calculated per training run according to Equation 45. The success threshold for the temporal loss metric is set at $L_T \leq 0.01$.

$$w \sim \mathcal{N}_{trunc}(\mu = 0, \sigma = \sigma_c) \qquad (44)$$

$$L_T = D(\pi(\mathbf{s}_t), \pi(\mathbf{s}_{t+1})) = \sum_{t=1}^{N} ||\mathbf{a}_{t-1} - \mathbf{a}_t)||_2 \qquad (45)$$

Looking at the results in Table 5, the success rates for three critic standard deviations can be seen with $\sigma_c = 0.05$ the nominal case. The convergence rate tracks the number of runs that become unstable in the parameters, while the total success rate combines the convergence and temporal metrics. A lower standard deviation results in a more stable training experience, indicated by the 100% success rate of a lower standard deviation compared to the nominal case. A higher standard deviation then results in a lower success rate.

To show the effect of random critic initialization on the response, the interquartile range of the states over 50 runs of the nominal case is presented in Figure 12. Only the successful runs are presented meaning 72% or 36 runs. This shows the level of variation the random seed causes during the training task. This shows a satisfactory and safe training response with the states remaining close to the reference signal, even during the initial convergence period before t=20s, but keeping in mind the success rate.

Due to the hybrid policy design and identity initialization of online learning policy layers, it is proposed that a failed online training run can safely be reset due to the presence of the pre-trained SAC policy layers. This is in contrast with IDHP-only methods which start with zero knowledge at the start of the training phase.

Table 5 Success rates on SAC-IDHP attitude training task with varying random critic initialization. Analysed over 50 runs per configuration.

Network Initialization	Convergence Rate	Temporal Loss Rate	Total Success Rate
$\sigma_c = 0.01$	100.0%	100.0%	100.0%
$\sigma_c = 0.05$ (nominal)	92.00%	78.26%	72.00%
$\sigma_c = 0.10$	68.00%	70.59%	48.00%

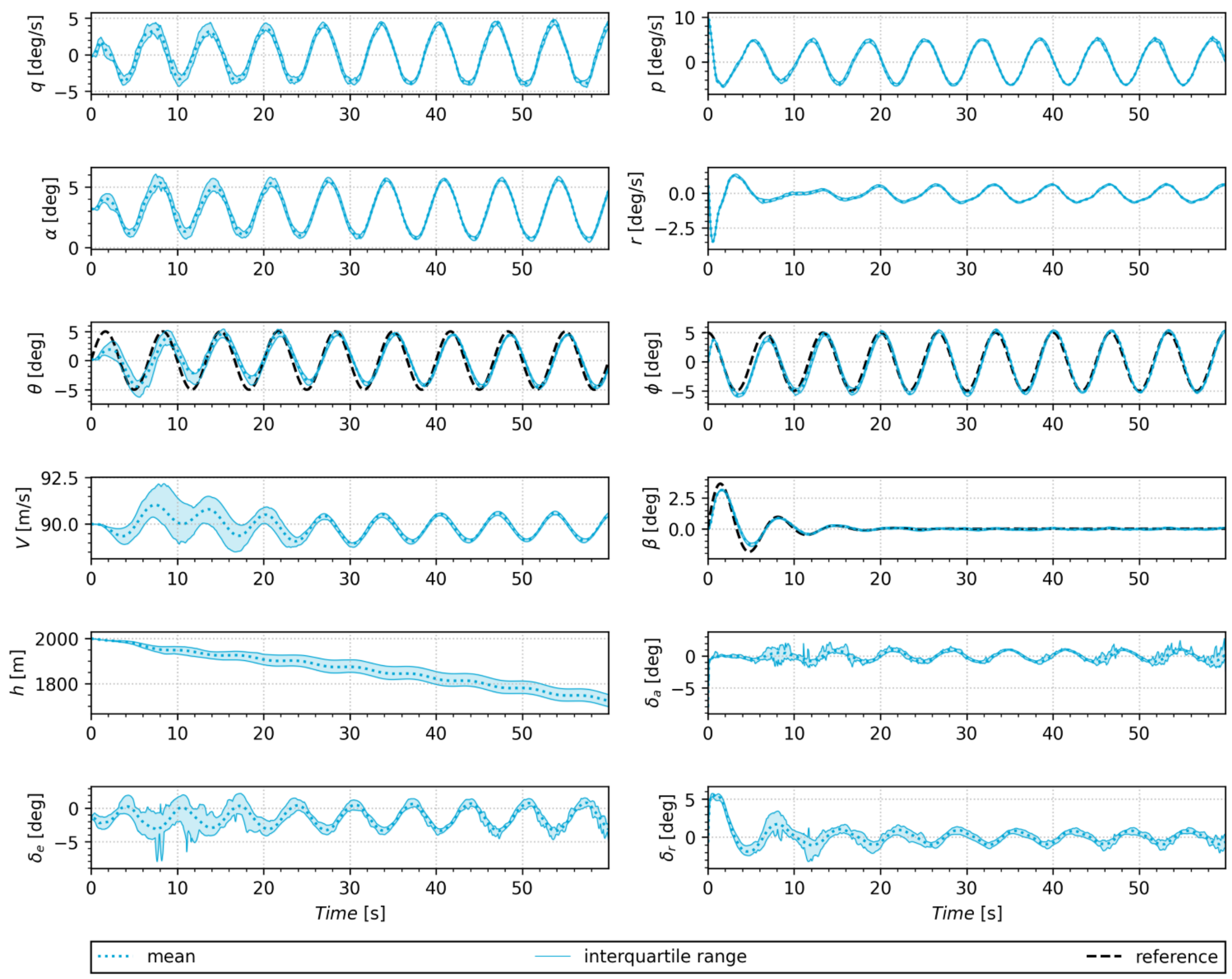


Fig. 12 SAC-IDHP response on attitude training task with $\sigma_c = 0.05$ over 36 successful runs.

V. Conclusion

It is demonstrated that a hybrid SAC-IDHP offline-online learning controller can be successfully implemented and provide coupled-dynamics fault-tolerant flight control in a complete RL-based cascaded altitude controller. It is shown that compared to SAC-only, the online learning of the hybrid policy architecture provides more adaptive control with lower tracking error across all tested nominal and failure cases. An improvement in nMAE of 0.74%, 5.46% and 0.82% is demonstrated for nominal case, longitudinal and lateral failure cases respectively. Compared to IDHP-only, random initialization of the actor is removed by the hybrid policy. It is proposed that this provides the ability to revert to a robust response only and reset online IDHP learning safely in flight due to the presence of robust pre-trained policy layers. Additionally, the hybrid architecture provides increased confidence in including IDHP into a fully coupled-dynamics 6-degree-of-freedom control loop.

The SAC-IDHP agent however exhibits increased oscillations mainly in the longitudinal states, most noticeable when adding biased sensor noise, but still providing lower tracking error compared to SAC-only. Challenges with offline SAC learning remain, including inconsistent training performance due to the many stochastic factors, but is improved by the use of CAPS regularization. Further research is recommended into comparison with IDHP-only and the safety of SAC and IDHP in covering the entire flight envelope before executing flight tests on the PH-LAB aircraft.

References

[1] Cokorilo, O., "Urban Air Mobility: Safety Challenges," *Transportation Research Procedia*, Vol. 45, 2020, pp. 21–29. https://doi.org/10.1016/j.trpro.2020.02.058.

[2] "Loss of Control In-Flight Accident Analysis Report 2019 Edition," *International Air Transport Association*, 2019, p. 44.

[3] Balas, G. J., "Flight Control Law Design: An Industry Perspective," *European Journal of Control*, Vol. 9, No. 2, 2003, pp. 207–226. https://doi.org/10.3166/ejc.9.207-226.

[4] Richard S. Sutton, and Andrew G. Barto, *Reinforcement Learning, Second Edition : An Introduction*, Adaptive Computation and Machine Learning, Vol. Second edition, Bradford Books, Cambridge, Massachusetts, 2018.

[5] Si, J., Barto, A. G., Powell, W. B., and Wunsch, D., *Handbook of Learning and Approximate Dynamic Programming*, Wiley-IEEE Press, 2004. https://doi.org/10.1109/9780470544785.

[6] Wang, F., Zhang, H., and Liu, D., "Adaptive Dynamic Programming: An Introduction," *IEEE Computational Intelligence Magazine*, 2009. https://doi.org/10.1109/MCI.2009.932261.

[7] Liu, D., Xue, S., Zhao, B., Luo, B., and Wei, Q., "Adaptive Dynamic Programming for Control: A Survey and Recent Advances," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 51, No. 1, 2021, pp. 142–160. https://doi.org/10.1109/TSMC.2020.3042876.

[8] 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. https://doi.org/10.2514/1.12597.

[9] 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. https://doi.org/10.1109/TNN.2003.813839.

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

[11] Zhou, Y., Van Kampen, E.-J., and Chu, Q., "Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control," *IMAV 2016*, Delft University of Technology, 2016.

[12] 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. https://doi.org/10.1016/j.conengprac.2017.12.011.

[13] 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*, AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, 2020. https://doi.org/10.2514/6.2020-1844.

[14] Kroezen, D., "Online Reinforcement Learning for Flight Control: An Adaptive Critic Design without Prior Model Knowledge," Master's thesis, Delft Unifersity of Technology, 2019.

[15] Lee, J., "Longitudinal Flight Control by Reinforcement Learning: Online Adaptive Critic Design Approach to Altitude Control," Master Thesis, Delft University of Technology, Delft, 2019.

[16] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., "Human-Level Control through Deep Reinforcement Learning," *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533. https://doi.org/10.1038/nature14236.

[17] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D., "Mastering the Game of Go without Human Knowledge," *Nature*, Vol. 550, No. 7676, 2017, pp. 354–359. https://doi.org/10.1038/nature24270.

[18] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P., "Trust Region Policy Optimization," *arXiv:1502.05477 [cs]*, 2015.

[19] 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," *arXiv:1509.02971 [cs, stat]*, 2019.

[20] 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)*, 2019, pp. 523–533. https://doi.org/10.1109/ICUAS.2019.8798254.

[21] Wang, Z., Li, H., Wu, Z., and Wu, H., "A Pretrained Proximal Policy Optimization Algorithm with Reward Shaping for Aircraft Guidance to a Moving Destination in Three-Dimensional Continuous Space," *International Journal of Advanced Robotic Systems*, Vol. 18, No. 1, 2021, p. 1729881421989546. https://doi.org/10.1177/1729881421989546.

[22] Shehab, M., Zaghloul, A., and El-Badawy, A., "Low-Level Control of a Quadrotor Using Twin Delayed Deep Deterministic Policy Gradient (TD3)," *2021 18th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2021, pp. 1–6. https://doi.org/10.1109/CCE53527.2021.9633086.

[23] He, L., Aouf, N., Whidborne, J. F., and Song, B., "Deep Reinforcement Learning Based Local Planner for UAV Obstacle Avoidance Using Demonstration Data," *arXiv:2008.02521 [cs]*, 2020.

[24] Cheng, Y., and Song, Y., "Autonomous Decision-Making Generation of UAV Based on Soft Actor-Critic Algorithm," *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 7350–7355. https://doi.org/10.23919/CCC50068.2020.9188886.

[25] Lee, M. H., and Moon, J., "Deep Reinforcement Learning-based UAV Navigation and Control: A Soft Actor-Critic with Hindsight Experience Replay Approach," *arXiv:2106.01016 [cs, eess]*, 2021.

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

[27] Dally, K., and Van Kampen, E.-J., "Soft Actor-Critic Deep Reinforcement Learning for Fault Tolerant Flight Control," *AIAA SCITECH 2022 Forum*, AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, 2021. https://doi.org/10.2514/6.2022-2078.

[28] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *arXiv:1801.01290 [cs, stat]*, 2018.

[29] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S., "Soft Actor-Critic Algorithms and Applications," *arXiv:1812.05905 [cs, stat]*, 2019.

[30] Mysore, S., Mabsout, B., Mancuso, R., and Saenko, K., "Regularizing Action Policies for Smooth Control with Reinforcement Learning," , May 2021. https://doi.org/10.48550/arXiv.2012.06644.

[31] Sun, B., and van Kampen, E.-J., "Incremental Model-Based Global Dual Heuristic Programming with Explicit Analytical Calculations Applied to Flight Control," *Engineering Applications of Artificial Intelligence*, Vol. 89, 2020, p. 103425. https://doi.org/10.1016/j.engappai.2019.103425.

[32] 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, 2017, pp. 493–496. https://doi.org/10.2514/1.G001762.

[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*, American Institute of Aeronautics and Astronautics, Kissimmee, Florida, 2018. https://doi.org/10.2514/6.2018-0385.

[34] van den Hoek, M. A., de Visser, C. C., and Pool, D. M., “Identification of a Cessna Citation II Model Based on Flight Test Data,” *4th CEAS Specialist Conference on Guidance, Navigation and Control*, 2017.

[35] Ba, J. L., Kiros, J. R., and Hinton, G. E., “Layer Normalization,” , Jul. 2016. https://doi.org/10.48550/arXiv.1607.06450.

[36] Kubota, S., Hayashi, H., Hayase, T., and Uchida, S., “Layer-Wise Interpretation of Deep Neural Networks Using Identity Initialization,” *arXiv:2102.13333 [cs]*, 2021.

[37] EASA, “CS-25 Amendment 27 - Review of Aeroplane Performance Requirements for Air Operations and Regular Update of CS-25,” https://www.easa.europa.eu/document-library/certification-specifications/cs-25-amendment-27, Jun. 2021.