A Multi-step and Eligibility Trace Approach to Incremental Dual Heuristic Programming for Flight Control

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

Incremental Dual Heuristic Programming (IDHP) is a successor to the Dual Heuristic Programming (DHP) algorithm that uses an online identified incremental system model, this algorithm showed promising flight control performance and tolerance of faults in simulation experiments. This paper studies the potential for extending IDHP through augmenting the computation of agent updates and returns, more specifically, by using eligibility trace updates and multi-step temporal difference error. This results in the MIDHP, IDHP(λ) and MIDHP(λ) algorithms, which are compared against IDHP in several simulated flight control scenarios with faults introduced mid-flight. The results demonstrate that the proposed algorithms have improved flight control performance and fault tolerance in terms of tracking errors when controlling a nominal aircraft and an aircraft with faults introduced, with the most improvement observed in MIDHP(λ).

Nomenclature

λ	=	Eligibility trace decay rate	[-]	ϵ, κ	=	RLS prediction residual, update gain	[-]
$\mathcal{S}, \mathcal{A}$	=	MDP state & action space	[-]	$\mathbf{E}$	=	Eligibility trace	[-]
$\mathcal{P}()$	=	MDP transition function	[-]	L_{CAPS}	=	CAPS score	[-]
n, m	=	No. of MDP states, actions	[-]	λ_T, λ_S	=	CAPS temporal, spatial smoothness weight	[-]
s, s', a, r	=	MDP state, augmented state, action, reward	[-]	$\mathcal{N}$	=	Multivariate normal distribution	[-]
γ	=	MDP reward discount rate	[-]	p, q, r	=	Roll, pitch, yaw rate	[rad/s]
R_t	=	MDP Return	[-]	V_{TAS}	=	True airspeed	[m/s]
$J(), \pi()$	=	Value, policy function	[-]	$\alpha, beta$	=	Angle of attack, sideslip	[rad]
Λ	=	Value gradient function	[-]	ϕ, θ, ψ	=	Roll, pitch, yaw angle	[rad]
Λ'	=	Target value gradient function	[-]	H	=	Height	[m]
W_a, W_c	=	Function parameter of actor, critic	[-]	X_e, Y_e	=	East, north-ward location	[m]
δ	=	Temporal difference error	[-]	θ_r	=	Reference pitch	[rad]
E_t	=	Quadratic δ	[-]	θ_e	=	Pitch error	[rad]
η_a, η_c	=	Actor, critic learning rate	[-]	$\delta_a, \delta_e, \delta_r$	=	Aileron, elevator, rudder deflection angle	[rad]
τ	=	Target critic update fraction	[-]	δ'_e	=	Elevator deflection from trim	[rad]
F, G	=	Incremental state transition, input matrix	[-]	tr_a, tr_e, tr_r	=	Aileron, elevator, rudder trim tab angle	[rad]

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

Θ	=	RLS parameter estimates	[-]	δ_f	=	Flaps deflection angle	[rad]
Σ	=	RLS parameter estimate covariance	[-]	T_1, T_2	=	Left, right engine thrust setting	[-]
X	=	RLS states	[-]	k	=	Learning rate factor	[-]
ρ	=	RLS forgetting factor	[-]	Sm	=	Action smoothness	[-]

I. Introduction

The civil aviation sector is undergoing many developments which will redefine what flight means, be it the advent of personal air vehicles, novel airliner designs, hydrogen fuelled concepts, or the increasing usage of drones [1–4]. Flight control on novel systems using existing control system design methods require rigorous and extensive modelling and system identification campaigns, costing a vast amount of time and resources. An example is in the gain-scheduling approach to flight control system design [5]. Furthermore, the occurrence of faults causes aircraft to fly outside the normal flight envelope, resulting in loss of control [6]. Such an issue is exacerbated by a reliance on model-based controller synthesis techniques, which focus on modelling nominal regions of the flight envelope. Fault tolerant and adaptive flight control is a trend in flight control design which directly addresses this issue, with promising control methods being actively developed [7–10].

Simultaneously, Reinforcement Learning (RL) has been developing at a rapid pace, from agents trained that can surpass human performance on games [11, 12], to ones with the ability to control real life systems [13, 14]. RL is a method of Machine Learning that predicates on the machine actively learning through sovereign actions, as opposed to other ML methods such as supervised learning where the machine is told what to do or what is correct. Actor-Critic Design (ACD) is a sub-field of RL which approaches the problem of RL from an optimal control perspective [15], where an agent comprises of an actor, a critic, and a system model, the former two modelled using function approximators such as a neural network, and latter modelled as state derivatives. The actor and critic are updated through steps known as policy improvement and policy evaluation respectively. ACD algorithms are generally classified into three categories, all of which have identical architecture for the actor but not the critic: Heuristic Dynamic Programming (HDP), where the critic estimates the actor's value function; Dual Heuristic Programming (DHP), where the critic estimates the actor's value function gradient; and Global Dual Heuristic Programming (GDHP), whose critic estimates both simultaneously [16]. Incremental DHP (IDHP) is a successor to the DHP algorithm, which is extended with an online identified incremental model of the system dynamics to facilitate agent updates [17]. Parallel to ACD, there also exist Deep RL (DRL) algorithms where deep neural networks are used to model the agent. However, the present paper will focus on ACD algorithms.

The intersection of these two developments has several promising potentials. The learning emulation that RL methods offer has the potential of making flight controllers truly intelligent, and offers an alternative reward-driven adaptive control method to methods such as Incremental Nonlinear Dynamic Inversion (INDI) or Incremental Back Stepping (IBS). Such a reward-driven adaptive control method can be directed to perform more than only performance tracking, which methods such as INDI and IBS are designed to do. In fact, RL controllers are an approach akin to the Linear Quadratic Regulator (LQR), to which ACD algorithms are closely related. Moreover, challenges in novel aircraft systems and fault tolerant control could be overcome with the successful application of RL based flight controllers, which can learn to fly an aircraft without any prior model. This model agnostic nature further implies the ability to use such controllers to fly systems which vary over time. Or more importantly, experience faults during operation, e.g. a bird strike event on an aircraft mid flight. ACD algorithms are especially suited for this role due to their high adaptiveness and high sample efficiency, allowing them to learn a stabilizing controller online during flight [17]. Success in the application of RL to flight control with faults introduced has been demonstrated in attitude control of the Innovative Control Effectors (ICE) model with a pure ADP controller [18] and an NDI hybrid ACD controller [19], as well as in attitude and velocity control of a business jet using a pure ACD controller [20].

ACD algorithms traditionally use information from one timestep per agent update. However, by using information from more time steps it is possible to speed up agent learning. This can be done in two ways, by using eligibility traces where past function parameter updates are recorded and subsequently reused [21], or by using observed rewards from

multiple time steps resulting in multi-step policy evaluations or updates [22]. These ideas have been applied to the HDP and GDHP algorithms with success, improving their learning rate [21–24]. With improved learning rates, the speed at which a controller made from such algorithms recovers from sudden system faults could be improved as well. Thus, multi-step and eligibility traces for ACD are interesting avenues to explore for fault tolerance RL based controllers. Additionally, such techniques have yet to be applied to IDHP.

Therefore, this paper's main contribution is in the extension of IDHP using eligibility trace updates, resulting in IDHP(λ), using the multi-step policy evaluations or Temporal Difference (TD) error, resulting in MIDHP, and a combination of the two, resulting in MIDHP(λ). These algorithms are evaluated both during nominal flight and with faults introduced to give an insight into these augmentation's effects.

This paper is outlined as follows: in Sec. II the background on IDHP, the application of IDHP to flight control, and the idea of multi-step updates and eligibility traces are presented; in Sec. III the proposed methodology for incorporating multi-step updates and eligibility traces to IDHP and the flight control task devised for testing are presented. The main results are presented and discussed in Sec. IV, with the main conclusions drawn in Sec. V.

II. Background

This section presents the background underlying the methodology of this paper. Beginning with introducing the sequential decision making framework used in all RL methods, then introducing IDHP and the sub-field of ACD in which it exists, and finally onto the eligibility trace and multi-step augmentations which exist in the TD learning framework.

A. MDP

Markov Decision Process (MDP) is the mathematical framework used in RL to model the sequential decision making process of some agent interacting with an environment. An MDP is described by the state space $\mathcal{S} \subset \mathbb{R}^n$, the action space $\mathcal{A} \subset \mathbb{R}^m$, a reward signal $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a state transition function $\mathcal{P}(s_{t+1}, r_{t+1}|s_t, a_t)$ which may deterministically or stochastically transition the state from s_t to s_{t+1} with reward r_{t+1} given an agent action a_t.

An MDP operates in discrete time. At each time step, an action is performed by the agent on the environment, which the agent responds to by producing the next time step's state and reward, which are both observed by the agent and influence subsequent agent decisions. This notion of sequential decision making has important ramifications. It implies that decisions made at an earlier time may influence what decisions should be made at a later time. In practice, this notion implies that taking suboptimal actions at an earlier time may lead to fewer rewards in the future. Building upon the MDP framework, it is possible to define a *Value function* $J(s)$, which describes the total future reward to be expected by the agent for starting in state s. The field of RL is dedicated to estimating such a function and exploiting it to the fullest degree.

B. ACD and IDHP

RL agents seek to maximize the total expected reward in a given MDP, that is to not only obtain the maximum reward currently but also the maximum reward in the future. There are several methods of creating such an agent, including ACD algorithms, which this paper will focus on. Such algorithms are composed of three components; the actor, which observes the state of the MDP and chooses an action it thinks is most optimal; the critic, which observes the state of the MDP and outputs either the value function, value function's gradient, or a combination of the two, all capturing information regarding the expected rewards from the current state; and the model, which models the dynamics of the MDP states. Function approximators are used to approximate the true functions which underly each of these components, for example, the true value function. Common options for such approximators are linear functions such as polynomials, radial basis functions, and neural networks.

In traditional ACD, either an offline phase for learning the MDP dynamics or knowing the dynamics a-priori were the two main techniques for applying ACD algorithms [20, 25, 26], as opposed to purely online learning. For pure online learning, there is a need to remove the burden of knowing system dynamics a-priori, and to efficiently identify a usable dynamics model online. In answer to this need, an incremental model identified online using Recursive Least

Squares (RLS) was proposed, resulting in IDHP [17]. Such a model came at the cost of only being able to model local dynamics as linear, as opposed to identifying nonlinear dynamics across the entire state space, which furthermore required high enough model update rates to overcome nonlinearities in system dynamics. However, this also has the added benefit of significantly reducing the complexity of model identification, allowing for computationally efficient model identification using a least squares approach.

The IDHP actor, also referred to as the policy, is a function denoted as $\pi : \mathcal{S} \rightarrow \mathcal{A}$. To act, the agent evaluates the policy $\pi()$ and obtains the appropriate a, which is then performed by the agent on the environment. Any function approximator with parameters W_a may be used to represent $\pi()$. To train the agent to maximize future expected reward, the parameters W_a are updated to maximize the return $R_t = r_t + \gamma J(s_{t+1})$ which is an estimate of the total reward to be expected in the future, made up of the current reward r_t and $J()$ discounted by $\gamma \in [0, 1]$. In ACD, updating W_a to maximize R can then be done using gradient ascent with the following Equations:

$$W_{a,t} = W_{a,t-1} + \eta_a \frac{\partial R_{t-1}}{\partial W_a} \tag{1}$$

$$\frac{\partial R_t}{\partial W_a} = \left(\frac{\partial r_t}{\partial s_{t+1}} + \Lambda_{t+1} \right) \frac{\partial s_{t+1}}{\partial a_t} \frac{\partial a_t}{\partial W_a} \tag{2}$$

$$\Lambda_t = \frac{\partial J(s_t)}{\partial s_t} \tag{3}$$

Where η_a is the gradient descent step size, also referred to as the actor's learning rate.

The IDHP critic is a value gradient function denoted as $\Lambda : \mathcal{S} \rightarrow \mathbb{R}^n$, it outputs the gradient of the value function with respect to the MDP states s as expressed in Eq. 3. This function is internal to the agent and serves only to aid in improving the actor's actions. Similar to the actor, any function approximator with parameters W_c may be used to represent Λ. To stabilize learning, a target critic denoted as Λ' is maintained, though IDHP can function without it. This target critic is identical to the critic except for the function parameters used, which are updated with a slow moving average filter towards the latest critic weights [27], see Eq. 7. The objective of the critic is to accurately estimate the value function's gradient. This is done by updating the critic's parameters to minimize a quadratic E_t of the TD error δ_t, where $E_t = \frac{1}{2}\delta_t \cdot \delta_t$ with $\cdot$ being the dot product, and δ_t defined as follows:

$$\delta_t = \Lambda_{t-1} - \frac{\partial r_{t-1}}{\partial s_{t-1}} - \gamma \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}} \tag{4}$$

Note that the target critic output is used in constructing δ_t, which serves the purpose of smoothing the variation of δ_t over time, as the target critic is updated slowly. In ACD, W_c is updated to minimize E through gradient descent using Eq. 5.

$$W_{c,t} = W_{c,t-1} - \eta_c \frac{\partial E_{t-1}}{\partial W_c} \tag{5}$$

$$\begin{aligned} \frac{\partial E_t}{\partial W_c} &= \frac{\partial E_t}{\partial \delta_t} \frac{\partial \delta_t}{\partial \Lambda_t} \frac{\partial \Lambda_t}{\partial W_c} \\ &= \delta_t \frac{\partial \Lambda_t}{\partial W_c} \end{aligned} \tag{6}$$

$$W_{c',t+1} = \tau W_{c,t} + (1 - \tau) W_{c',t} \tag{7}$$

The system model of IDHP is then used to identify the $\frac{\partial s_{t+1}}{\partial a_t}$ term from Eq. 2 and the $\frac{\partial s_t}{\partial s_{t-1}}$ term from Eq. 4. This model is also internal to the agent, it is used to improve the actions of the actor and to improve the value estimation of the critic. Assuming that a discrete model of the system exists, an incremental form of the discrete model can be written as $\Delta s_{t+1} = F_t \Delta s_t + G_t \Delta a_t$. This results in the following definition for the two partial derivatives

$$\frac{\partial s_{t+1}}{\partial a_t} \approx G_t$$

$$\frac{\partial s_{t+1}}{\partial s_t} \approx F_t + G_t \frac{\partial a_t}{\partial s_t} \tag{8}$$

Using RLS, a linear model can be constructed by first defining the state increment $\delta s_{t+1} = s_{t+1} - s_t$ and action increment $\delta a_{t+1} = a_{t+1} - a_t$, then defining the linear model parameter matrix $\Theta_t = \begin{bmatrix} F_t & G_t \end{bmatrix}^\top \in \mathbb{R}^{(n+m)\times n}$ and variable vector $X_t = \begin{bmatrix} \delta s_{t+1} & \delta a_{t+1} \end{bmatrix} \in \mathbb{R}^{n+m}$. With the RLS algorithm, Θ_t can then be identified online according to Alg. 1.

Algorithm 1 exponentially weighted RLS algorithm.

1: **Initialize**. Parameter covariance $\Sigma_0 \in \mathbb{R}^{(n+m)\times(n+m)}$, parameter $\Theta_0 \in \mathbb{R}^{(n+m)\times n}$.
2: **Online Model Identification**. For each time step, $t = 1, 2, \cdots$, compute:

$$\begin{aligned}
\delta s_t &= s_t - s_{t-1} \\
\delta a_t &= a_t - a_{t-1} \\
X_t &= \begin{bmatrix} \delta s_t & \delta a_t \end{bmatrix}^\top \\
\kappa_t &= \frac{\Sigma_{t-1} X_t}{\rho + X_t^\top \Sigma_{t-1} X_t} \\
\epsilon_t &= \delta x_t - X_t^\top \Theta_{t-1} \\
\Theta_t &= \Theta_t + \kappa_t \epsilon_t \\
\Sigma_t &= \frac{1}{\rho}(\Sigma_{t-1} - \kappa_t X_t^\top \Sigma_{t-1})
\end{aligned}$$

C. Multi-step Updates and Eligibility Traces

The TD error computes the error between the value which the critic estimated and the value which is observed, this observed value is referred to as the TD target. In the case of Eq. 4, the TD target is composed of the terms $\frac{\partial r_{t-1}}{\partial s_{t-1}} + \gamma \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}}$. It can be seen that the TD target is composed of an observed reward signal and an estimate made by the target critic. The idea of multi-step TD error is to use more observed reward signals to construct a more accurate TD target [28, 29].

For the IDHP's TD error, a multi-step version is expressed as follows:

$$\delta_{n,t} = \Lambda_{t-n} - \frac{\partial(\gamma^n J'_t + \sum_{m=0}^{n-1} \gamma^m r_{t-n+m})}{\partial s_{t-n}} \tag{9}$$

Instead of using the reward at one single time step to compute the TD error, it is possible to extend the error to include reward from multiple time steps

An alternative and more general method for incorporating additional information to agent updates is the eligibility trace **E**, which keeps track of how eligible each function parameters are in each update step. This is done by storing previous function gradients and reusing them in the future at decaying magnitudes [30]. The function gradient in question is the derivative of the function output with respect to each function parameter, for the actor's case **E** would accumulate the $\frac{\partial a_t}{\partial W_a}$ term from Eq. 2, and for in the critic's case **E** would accumulate $\frac{\partial \Lambda_t}{\partial W_c}$ from Eq. 6. In implementation, the eligibility trace results in a similar algorithm as momentum gradient descent, which accumulates past function updates and also reuses them in the future at decaying magnitude. The difference between these two methods is in the variable being accumulated. Whereas momentum gradient descent accumulates the total parameter update, eligibility traces only accumulate the function's gradient.

To incorporate eligibility traces, the parameter update equations are changed to the form shown in the following equations:

$$W_t = W_{t-1} + \eta \frac{\partial M}{\partial O} \mathbf{E}_{t-1} \tag{10}$$

$$\mathbf{E}_t = \lambda\gamma \mathbf{E}_{t-1} + \nabla W_{t-1}, \quad \mathbf{E}_0 = \mathbf{0} \tag{11}$$

Where ∇W is the gradient of the network output with respect to network weights, $\frac{\partial M}{\partial O}$ is the gradient of the relevant Metric with respect to network Output, for the critic this would be δ_t and for the actor this would be $\left(\frac{\partial r_t}{\partial s_{t+1}} + \Lambda_{t+1}\right)$; $\lambda \in [0, 1)$ is the trace decay rate which controls how quickly do prior gradients decay in the eligibility trace, $\lambda = 0$ decays previous gradients completely and results in the original update rules, while $\lambda \to 1$ means previous gradients decay according to γ.

A multi-step policy evaluation has been successfully augmented to the HDP algorithm [22] which yielded increased sample efficiency, an improvement further refined by an adaptive multi-step scheme [23]. An eligibility trace style update has also been applied successfully to ACD algorithms on several works [21, 24, 31], such an augmentation similarly improved the agent's sample efficiency.

III. Methodology

Taking the idea of multi-step updates and eligibility traces from TD learning, it becomes possible to refine the method of updating the actor and the critic in the ACD algorithms. The methods for incorporating such ideas into the IDHP algorithm, and how the augmentations are to be evaluated, are outlined in the methodology presented in this section.

A. IDHP Augmentations

Multi-step updates and eligibility traces can be incorporated individually or together in the IDHP algorithm, with multi-step TD applied to the critic and eligibility traces applied to either the critic or the actor. However, from preliminary empirical testing, the option of applying eligibility traces to the critic was shown to negatively impact the learning performance of IDHP. Thus the three options present are to: apply multi-step updates to the critic, apply eligibility traces to the actor, or the two options combined.

For the critic, which is only augmented with the multi-step TD error, the gradient descent term of Eq. 6 used for updating the critic is reworked to include a multi-step TD error. While each algorithm or MDP being solved typically has a certain n which is optimal, the present paper only considers a 2-step TD for the sake of simplicity, which can be obtained by substituting $n = 2$ to Eq. 9, which yields $\delta_{2,t}$ expressed in Eq. 12. To incorporate the 2 step TD error into the critic update is to simply substitute δ_t with $\delta_{2,t}$ in Eq. 6.

$$\begin{aligned}
\delta_{2,t} &= \Lambda_{t-2} - \frac{\partial(\gamma^2 J'_t + \sum_{m=0}^{1} \gamma^m r_{t-2+m})}{\partial s_{t-2}} \\
&= \Lambda_{t-2} - \frac{\partial(\gamma^2 V'_t + r_{t-2} + \gamma r_{t-1})}{\partial s_{t-2}} \\
&= \Lambda_{t-2} - \gamma^2 \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}}\bigg|_t \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1} - \frac{\partial r_{t-2}}{\partial s_{t-2}} - \gamma \frac{\partial r_{t-1}}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1}
\end{aligned} \tag{12}$$

Regarding the actor, the primary augmentation is in incorporating eligibility traces. To do so, the gradient descent term $\frac{\partial a_t}{\partial W_a}$ of Eq. 2 is replaced by $\mathbf{E}$, which is updated according to Eq. 13. A secondary augmentation is to add a Conditioning for Action Policy Smoothness (CAPS) parameter to mitigate the noisey actions commonly observed in RL agents [32]. This is through L_{CAPS} as defined in Eq. 14 which penalizes large spatial and TD in the actor's actions, the weights of these penalties can be adjusted through the factors λ_S and λ_T respectively. Combining these two augmentations results in a new expression shown in Eq. 15 for the gradient descent term $\frac{\partial R_t}{\partial W_a}$ used in Eq. 1.

$$\mathbf{E}_t = \lambda\gamma\mathbf{E}_{t-1} + \frac{\partial a_t}{\partial W_a} \tag{13}$$

$$L_{CAPS} = \lambda_T \|\pi(s_t) - \pi(\tilde{s})\|_2 + \lambda_S \|\pi(s_t) - \pi(s_{t-1})\|_2 \,, \quad \tilde{s} \sim \mathcal{N}(s_t, diag(\sigma)) \,, \quad \sigma \in \mathcal{R}^n \tag{14}$$

$$\frac{\partial R_t}{\partial W_a} = \left(\left[\frac{\partial r_t}{\partial s_{t+1}} + \Lambda_{t+1} \right] \frac{\partial s_{t+1}}{\partial a_t} + L_{CAPS} \right) \mathbf{E}_t \tag{15}$$

Ultimately, this results in 3 augmented IDHP algorithms in addition to the baseline IDHP algorithm. The first augmented algorithm is IDHP(λ), which has the actor augmented with an eligibility trace update; the second is MIDHP, which has the critic augmented with a multi-step update; the third is MIDHP(λ), which has both the two augmentations simultaneously.

B. Aircraft Pitch Control as an MDP

The four algorithms are evaluated in their ability to control the pitching motion of an aircraft in a flight control task. This is done in simulation, where a high fidelity nonlinear 6 degrees of freedom dynamics model of a fixed-wing business jet running at 100 Hz or $dt = 0.01$ s is used [33]. The aircraft is modelled with engine and control surface dynamics, such as inertia in deflecting the control surfaces. Airspeed control is delegated to an auto-throttle controlling engine thrust settings. The aircraft model has been validated by flight test data onboard a Cessna Citation II research aircraft [34]. During the experiments conducted in this paper, the aircraft model's state and inputs are trimmed for an airspeed of 90 m/s at an altitude of 2000 m.

The pitch control task needs to first be framed in terms of an MDP before the IDHP algorithm is applied as a flight controller. The control scheme which the MDP will emulate is a simple feedback control scheme, which can be represented in the form of Figure 1.

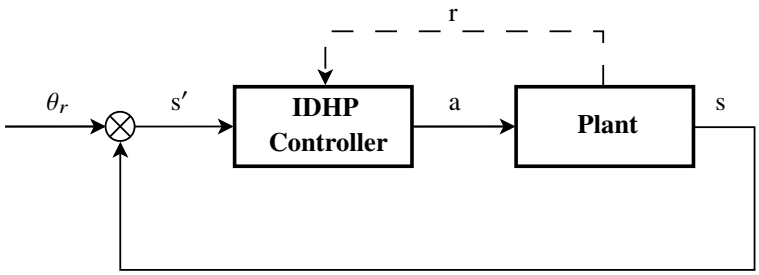


Fig. 1 Control configuration of the pitch control MDP.

The dynamics model of the aircraft already fulfils the role of the state transition function $\mathcal{P}(s_{t+1}, r_{t+1}|s_t, a_t)$, this model is the plant used in Fig 1. Thus, the only remaining variables to fully define the MDP are s, a, and r. In addition to these variables, $\frac{\partial r}{\partial s}$ also needs to be defined for the IDHP algorithm,

The aircraft model used has a state vector x as shown in Eq. 16. To create s, only a subset of x states will be used, causing the resultant MDP being only partially observable (POMDP). Specifically, s is made to include only a select number of longitudinal aircraft states, as shown in Eq. 17. In addition, to make the control task more explicit to the agent, the actor and critic functions are provided with the pitching error $\theta_e = \theta - \theta_r$. This addition requires a slight abuse of notation where the actor and critic inputs will be the augmented MDP state s' shown in Eq. 18, instead of only s.

$$x = \begin{bmatrix} p & q & r & V_{TAS} & \alpha & \beta & \theta & \phi & \psi & H & X_e & Y_e \end{bmatrix}^\top \tag{16}$$

$$s = \begin{bmatrix} \alpha & \theta & q \end{bmatrix}^\top \qquad s' = \begin{bmatrix} \alpha & \theta & q & \theta_e \end{bmatrix}^\top \tag{17, 18}$$

The full set of actuators modelled u is shown in Eq. 19. Just like for s, not all of the modelled variables will be used in creating a. The agents in this control task will only be in control over the elevator, specifically the elevator deflection from the trim angle δ'_e, see Eq. 20. In addition, a symmetric deflection limit close to the realistic saturation limits of the PH-LAB is set on the elevator δ_e.

$$u = \begin{bmatrix} \delta_e & \delta_a & \delta_r & \text{tr}_e & \text{tr}_a & \text{tr}_r & \delta_f & \text{gear} & T_1 & T_2 \end{bmatrix}^\top \tag{19}$$

$$a = \begin{bmatrix} \delta'_e \end{bmatrix}, \quad -15\frac{\pi}{180} < \delta'_e < 15\frac{\pi}{180} \text{ [rad]} \tag{20}$$

The reward r is defined as the negative square of the pitch tracking error, as shown in Eq. 21, and it's derivative $\frac{\partial r}{\partial s}$ shown in the following:

$$r = -(\theta - \theta_r)^2 \qquad \frac{\partial r}{\partial s} = \begin{bmatrix} 0 & -2(\theta - \theta_r) & 0 \end{bmatrix} \tag{21, 22}$$

At initialization, the dynamics model states and inputs are set equal to the trim values, according to the values as follows:

$$x_0 = \begin{bmatrix} 0 & 0 & 0 & 90 \text{ m/s} & 0.0576 \text{ rad} & 0 & 0 & 0.0576 \text{ rad} & 0 & 2000 \text{ m} & 0 & 0 \end{bmatrix}^\top \tag{23}$$

$$u_0 = \begin{bmatrix} -0.02855 \text{ rad} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55\,\% & 55\,\% \end{bmatrix}^\top \tag{24}$$

C. IDHP Algorithm Configuration

The constants and variables which need to be initialized for the IDHP algorithm, including the dynamic model state and inputs of the MDP, are tabulated in Tab. 1. The version of IDHP implemented uses varying learning rates and eligibility trace decay rates, starting at higher values during the warmup phase, and then decaying to lower values during the manoeuvring phase. These high and low values are denoted by an h and l in the subscript respectively, specifically $\eta_{\cdot,h}$, $\eta_{\cdot,l}$, λ_h, and λ_l. Instead of transitioning the learning and decay rates abruptly between the warmup and manoeuvering phases, a continuous transition is done instead. At the end of the warmup, the learning and decay rates are multiplied by a factor k_t defined at every t as follows:

$$k_t = c + (1 - c)\frac{v_l}{v_t}, \quad c = 0.998 \tag{25}$$

Where v_t is the rate's value at t, and v_l is the lower value of the rate. The constant c changes how rapidly the rate drops, and is set to be 0.998 for the present study. Multiplying v_t by the factor k_t brings v_t asymptotically towards v_l, that is to say $\lim_{t\to\infty} v_h \prod_{i=0}^{t} k_i = v_l$.

Table 1 IDHP initialization variables and hyperparameters.

Hyperparameters		Variable initialization			
Actor learning rates:	$\eta_{a,h}$,$\eta_{a,l}$	Eligibility trace:	$\mathbf{E}_0$	=	$\mathbf{0}$
Critic learning rates:	$\eta_{c,h}$,$\eta_{c,l}$	RLS model parameter:	Θ_0	=	$\mathbf{0}$
Eligibility trace decay rates:	λ_h,λ_l	RLS model covariance:	Σ_0	=	$10^6 \cdot \mathbb{I}$
MDP reward discount factor:	$\gamma = 0.6$	Actor-network weights:	$W_{a,0}$	$\sim$	$\mathcal{N}(\mathbf{0}, 0.1^2\mathbb{I})$
RLS forgetting factor:	$\rho = 1$	Critic-network weights:	$W_{c,0}$	$\sim$	$\mathcal{N}(\mathbf{0}, 0.1^2\mathbb{I})$
Target critic mixing factor:	$\tau = 0.02$	Trim states:	x_0	=	Eq 23
		Trim inputs:	u_0	=	Eq 24

For the function approximators of the actor and the critic, a single hidden layer neural network with 10 nodes in the hidden layer is used. This network size is a compromise struck between learning complexity and agent performance, as it was observed that this size is just before agent performance degrades, but not too big that the number of parameters to be optimized slows down learning. In both networks, there are no biases in any of the layers, the input layer takes s' and has four nodes, and the hidden layer uses a tanh activation function. The output layer of the two networks are different. For the actor, its output is elevator deflection from trim δ'_e and so only a single node in the output layer is required, a tanh activation function is also used in this layer with the upper and lower asymptotes being $15\pi/180$, the limits stated in Eq. 20. For the critic, its output layer estimates value function gradients over s and thus has 3 nodes, a simple linear activation function is used.

The IDHP algorithm is created by combining the three modules and defining the flow of variables from one module to another. This is depicted graphically in Figure 2, the inputs of the IDHP agent are r, s, and θ_r, while the output is a. IDHP uses the inputs for two categories of computation, the first of which is to compute the control action, which is done simply by the actor, and the second category is to compute all the necessary internal updates. The order of update computations is clarified further in Alg. 2.

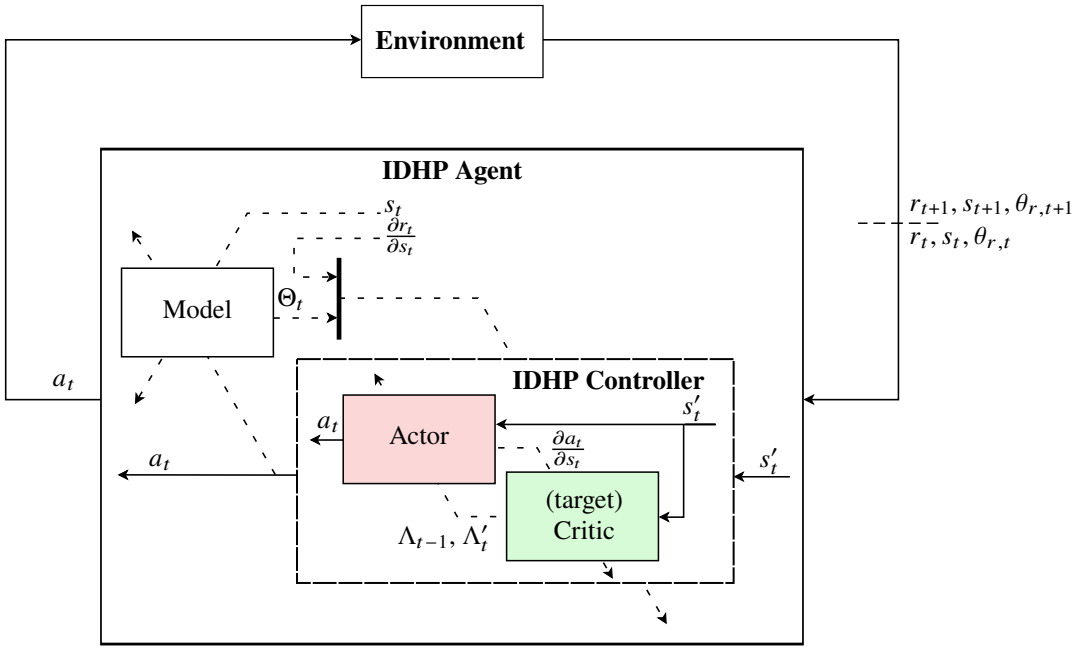


Fig. 2 MDP flow diagram representing the signal flows internal to the IDHP agent, dashed signals represent variables used to update the blocks which they cross.

Algorithm 2 IDHP algorithm.

1: **Initialize**:
Set initial variable values and hyperparameters listed in Tab. 1.
2: **Online loop**:
For $t = 0$ **to** T/dt **do**

$a_t \leftarrow \pi(s_t; W_{a,t})$ ▷ sample action from policy

$u_t = u_0 + a_t$ ▷ add commanded action to trim input

$s_{t+1}, r_{t+1} \leftarrow \text{Environment}(u_t)$ ▷ perform action to propagate dynamics

$$\eta_{a,t}, \eta_{c,t}, \lambda_t \leftarrow \begin{cases} \eta_{\cdot,h}, \lambda_h & \text{if} \quad t < t_{\text{warmup}} \\ \eta_{\cdot,t-1} k_t, \lambda_{t-1} k_t & \text{otherwise} \end{cases}$$ ▷ $k_t \leftarrow$ Eq. 25

If using multi-step IDHP **and** $t > 2$ **then**

$$\delta_t = \Lambda_{t-2} - \frac{\partial r_{t-2}}{\partial s_{t-2}} - \gamma \frac{\partial r_{t-1}}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1} - \gamma^2 \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}}\bigg|_t \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1}$$ ▷ Eq. 12, $\frac{\partial s_t}{\partial s_{t-1}} \leftarrow$ Eq. 8

Else

$$\delta_t = \Lambda_{t-1} - \frac{\partial r_{t-1}}{\partial s_{t-1}} - \gamma \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}}\bigg|_t$$ ▷ Eq. 4, $\frac{\partial s_t}{\partial s_{t-1}} \leftarrow$ Eq. 8

End If

$\frac{\partial E_t}{\partial W_c} = \delta_t \frac{\partial \Lambda_t}{\partial W_c}$ ▷ Eq. 6

$W_{c,t+1} = W_{c,t} - \eta_{c,t} \frac{\partial E_t}{\partial W_c}$ ▷ Eq. 5

$W_{c',t+1} = \tau W_{c,t} + (1 - \tau) W_{c',t}$

$\mathbf{E}_{t+1} = \lambda_t \gamma \mathbf{E}_t + \frac{\partial \pi(s_t)}{\partial W_a}$ ▷ Eq. 13, $\lambda = 0$ if not using eligibility traces

$\frac{\partial R_t}{\partial W_a} = \left(\left[\frac{\partial r_t}{\partial s_{t+1}} + \Lambda_{t+1} \right] G_t + L_{CAPS} \right) \mathbf{E}_t$ ▷ Eq. 15

$W_{a,t+1} = W_{a,t} + \eta_{a,t} \frac{\partial R_t}{\partial W_a}$ ▷ Eq. 1

$F_{t+1}, G_{t+1} \leftarrow$Algorithm 1

Implementation of the ∇W term in Eq. 11 differs based on the actor and critic function form used. In the case of the present IDHP, where eligibility traces are used only on the actor, whose function is a neural network, ∇W represents the derivative of the actor network with respect to its weights. Such a derivative is called the Jacobian matrix of the actor network, where the rows are the outputs of the network, and columns are the derivatives of the outputs with respect to each weight. For the actor, the Jacobian matrix is a 1 by 40 matrix, since in total the actor network has 1 output and $3 \cdot 10 + 10 \cdot 1$ weights, therefore $\nabla W = \frac{\partial \pi(s)}{\partial W_a} \in \mathcal{R}^{1 \times 40}$.

Since the augmented IDHP algorithms are either algorithmically different from the baseline, or have new hyperparameters, or both, each algorithm's hyperparameters are tuned individually to maximize that algorithm's performance. This leads to each algorithm having its own set of learning and decay rates, which are reported in Tab. 2.

Table 2 Hyperparameters used for each of the four algorithms.

		IDHP	IDHP(λ)	MIDHP	MIDHP(λ)
$\eta_{a,h}, \eta_{a,l}$	:	40, 10	25, 7.5	43, 6.5	30, 5.0
$\eta_{c,h}, \eta_{c,l}$	:	0.5, 0.25	0.5, 0.25	0.9, 0.05	0.7, 0.1
λ_h, λ_l	:	N/A	0.99, 0.8	N/A	0.99, 0.4

D. Experiment Setup

A simple flight manoeuvre is designed to evaluate the performance of the various algorithms, this manoeuvre is preceded by a warmup period where the agents learn a usable control policy to fly the aircraft. During the flight manoeuvre, several sudden faults will be introduced to the aircraft to evaluate the fault tolerance behaviour of the agents.

The overall flight lasts 90 s, the first 45 s are used to warm up the agents, followed by 10 s of rest for the aircraft states to settle back to trim, then the remainder of the flight the agents have to perform a simple pitch up and pitch down manoeuvre: representing actual automatic control in flight. Throughout the flight, the agents are to control the aircraft to follow a reference pitch signal θ_r which takes different shapes during the separate phases of flight. The signal θ_r first takes a sinusoidal form during warmup, a trapezoidal form during the pitch-up pitch-down manoeuvre, and remains equal to the θ_{trim} for all other times, this is described by Eq. 26. For convenience, the first 55 s of flight is referred to as the *warm up* phase, while the remainder is referred to as the *manoeuvring* phase.

$$\theta_{r,t} = \theta_{trim} + \frac{\pi}{180} \cdot \begin{cases} (10\sin(\frac{2\pi t}{15}) + 8\sin(\frac{2\pi t}{30}))(1 - \frac{t}{75}) & t < 45 \\ 1.5(t - 55) & 55 \leq t < 65 \\ 15 & 65 \leq t < 75 \\ -1.5(t - 85) & 75 \leq t < 85 \\ 0 & \text{otherwise} \end{cases} \quad \text{[rad]} \tag{26}$$

three fault cases are designed based on this 90 flight manoeuvre. The four algorithms will perform 100 flights over the same 100 different random number generator seeds on each fault case. Since the only stochasticity in the experiment is the initial actor and critic network weights, that will be the only difference between each run. Ultimately, the 100 runs will provide a more accurate view of each algorithm than simply running the algorithms once. The cases all introduce the corresponding faults around the beginning of the manoeuvring phase, at the 60 s mark of the flight. The first fault is a shift in the Centre of Gravity (CG) of the aircraft, where the longitudinal location of the CG is shifted forward by 0.5 m. The second fault is to be a damaged elevator which is modelled by multiplying the δ_e by 0.3, modelling how a damaged elevator reduces its control effectiveness. The third fault is a combined damaged elevator and reduced elevator deflection limits, where δ_e is multiplied by 0.3 and the maximum and minimum δ_e are set to be $\pm 5\frac{\pi}{180}$ rad.

The full time trace of the reference signal is shown in Figure 3. During each run which concerns a flight with faults introduced, it will be introduced at 60 s, the time indicated by the red cross.

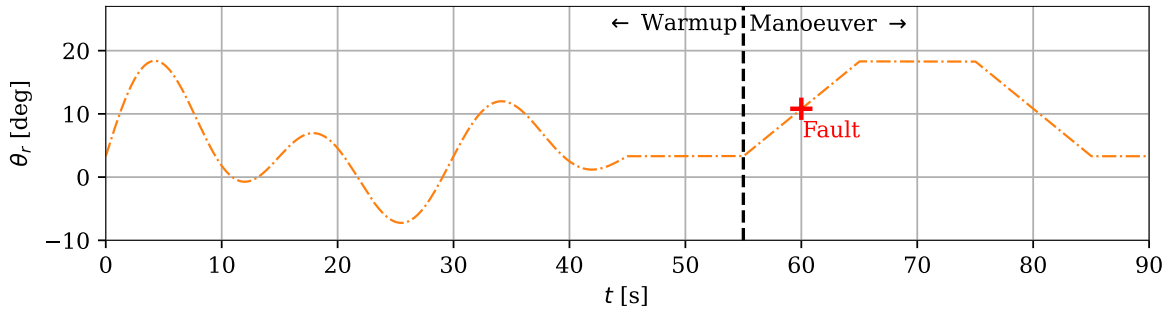


Fig. 3 Reference pitch signal θ_r, vertical black line at 55 s demarcates the end of the *warmup* phase and the start of the *manoeuvering* phase, red cross at 60 marks introduction of fault if any.

Five tests are compiled for the final evaluation and comparison of the algorithms, an overview of the tests are tabulated in Tab. 3. Test one studies the four algorithms during the warmup phase of the flight, while tests two, three, and four study how the four algorithms handle the nominal and faulty aircraft during the manoeuvering phase.

All five tests will use the same two metrics for algorithm evaluation, the first metric is the Root Squared Error (RSE) in θ_e, measuring the tracking accuracy of the controller and defined in Eq. 27. The second metric is a smoothness metric Sm, which measures controller action smoothness. It's definition is taken from [32] and presented in the following:

$$\text{RSE} = \sum_{t=0}^{T/dt} \sqrt{\theta_{e,t}^2} \qquad Sm = \frac{2}{nf_s} \sum_{i=1}^{n} M_i f_i \tag{27, 28}$$

Where M_i is the amplitude of the i-th frequency component f_i where $i \in [1, n]$, with n the number of frequency

components sampled, and $f_s = 100$ being the sampling frequency in time domain. Calculating Sm is done by first taking the Fourier transform of δ_e from the concerned flight phase, and then using the obtained spectrum in Eq. 28.

Table 3 The five tests used in evaluating the proposed augmentations on IDHP.

		Test 1	**Test 2**	**Test 3**	**Test 4**	**Test 5**
Phase	:	Warmup (0 to 55 s)	Manoeuvering (55 to 90 s)	Manoeuvering (55 to 90 s)	Manoeuvering (55 to 90 s)	Manoeuvering (55 to 90 s)
Faults	:	N/A	None	Shifted CG	Damaged Elevator	Damaged and saturated elevator
Metrics	:	RSE, Sm				

With the first test, the question of how the proposed augmentations affect the ability of IDHP to learn from tabular rasa may be answered. As the first test is concerned only with the performance of the controllers when learning from a randomly initialized state. The second to fifth tests will be the primary concern of this paper, they will tell of the fault tolerance behaviours of the four controllers as well as how they may perform in flight once trained.

E. Statistical Interpretation of Results

The five tests in Tab. 3 will generate samples of metrics for the four algorithms. To systematically decide which algorithm's metrics were better than another, two statistical measures must be used to interpret these results: one measure for quantifying the *statistical significance* of any difference, and another measure to quantify the *substantive significance* of such observed differences [35]. The first measure is the t-statistic from Student's t-test, which can be used to decide whether the mean of two samples' underlying distribution is different from another [36], under the critical assumption that the samples are normally distributed. The second measure is the A-value from Vargha and Delaney's A-test, this value can be used to rank the underlying distribution of two samples [37]. The A-value is bounded to the interval $[0, 1]$ and measures the relative difference of two samples, $A = 0.5$ means that there are no relative differences in the samples, < 0.5 means that the the first sample is smaller than the second, and vice versa for > 0.5.

While the A-value could be used to quantify the absolute difference between samples, the present paper will only use this measure for ranking different samples. As an example, this ranking can be done if two A-tests are taken where one sample is used in both tests, allowing for an ordinal ranking of the three samples involved in the two tests.

These statistics will be taken between the clean samples of IDHP and the augmented algorithms, that is to say, the gathered metrics are cleaned of outliers before taking any statistical tests. In such a manner, the following questions can be answered 1) whether an augmented algorithm's metrics are significantly different from IDHP's metrics, by reading the comparison's t-statistic, with a p-value of 0.05 being used to decide for statistical significance; and 2) by what magnitude this augmented algorithm's metrics are different from IDHP's metrics, through reading the A-value. Since all A-values are taken between IDHP and the augmented algorithm metrics, it would be possible to ordinally rank the metrics of the four algorithms for each test in Tab. 3. Summarily, the proposed augmentations' effects on the flight controllers may be concluded.

IV. Results and Discussion

The three proposed algorithms of IDHP(λ), MIDHP, and MIDHP(λ) along with the baseline algorithm IDHP are evaluated through the experiments designed in the Methodology Sec. III.D. The results gathered from these Monte Carlo experiments are presented and discussed in this section.

Before diving into the metrics of the five tests, the flight performance of the controller created by the IDHP algorithm with no augmentations and no faults introduced is presented in Figure 4, to provide some frame of reference for the metrics shown hereafter. The figure shows the flight of all 100 runs in translucent blue lines, with the aircraft pitch in the upper subplot and elevator deflection commanded by the IDHP controller in the lower subplot.

Figure 4 shows how the IDHP agent starts out not knowing how to control the aircraft, but over the first 10 s comes

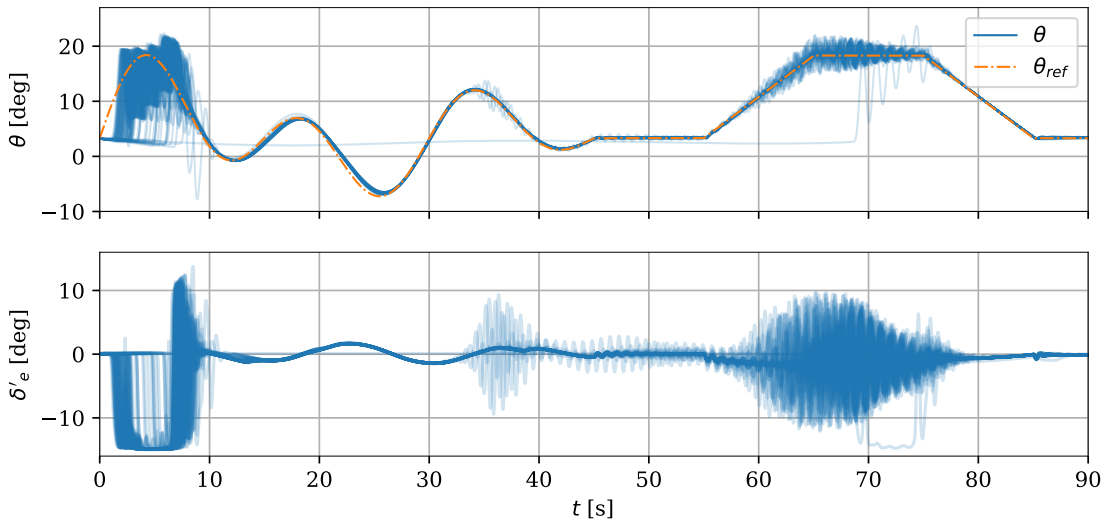


Fig. 4 IDHP controller flight time trace with no faults introduced to the aircraft.

to learn a stable and usable controller. The behaviour of this controller can be seen to stabilize over the remainder of the warmup phase, indicating that the actor-network weights have stabilized. Having warmed up the controller allowing it to learn a usable actor network, the aircraft is then commanded to trim for a short time before commencing the pitch-up pitch-down manoeuvre.

These samples of controllers can be seen to track the reference signal with good accuracy in portions of the flight. However, during the initial 10 s of the warmup and the maximum pitch-up segment of the manoeuvre, the controllers command a very oscillatory or noisy action. This occurs despite the CAPS parameter being incorporated into IDHP. Additionally, there are about 3 runs which exhibit oscillatory actions towards the end of the warmup phase around 35 s, which is symptomatic of a high gain controller. This issue becomes more stark during the manoeuvering phase, where at the high-pitch segment, many of the controllers become marginally stable, exhibiting highly oscillatory actions. Lastly, there is also one controller which did not manage to learn a stable controller during the warmup phase.

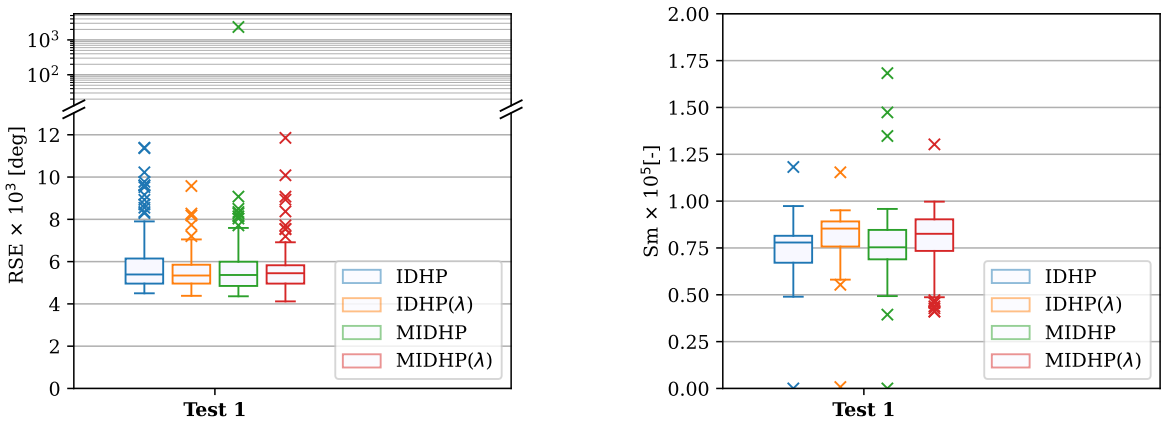
Taking a step back, the performances of the four algorithms over the five tests are presented in the following subsections using RSE and *Sm* metrics from the Monte Carlo experiments. These results are used to compare the augmented algorithms against the baseline IDHP.

A. Warmup Segment

The four algorithms' RSE and *Sm* metrics in Test 1, the test evaluating an algorithm's warmup performance, are shown in Figure 5. The statistical test results on the outlier-free or cleaned metric samples are summarized and presented in Tab. 4. From the figure, it appears that the tracking error and smoothness of all controllers are comparable, with the Inter-Quartile Range (IQR) being at similar values of around 5-6 $\times 10^3$ deg for RSE and 0.7-0.9 $\times 10^5$ for *Sm*.

The statistical testing results of Tab. 4 also tell a similar story. When it comes to RSE, there is no statistically significant difference between the four algorithms' controllers. However, a slightly smaller A-value is reported in all the comparisons between the augmented algorithms and the baseline IDHP, signifying a marginal yet statistically insignificant improvement in tracking performance by the augmentations during the warmup phase.

The *Sm* metric statistical tests show that there is statistically significant differences between the smoothness of the augmented algorithms of IDHP(λ), and MIDHP(λ), versus the baseline IDHP. Specifically, both these two augmented algorithms demonstrated a less smooth control action during warmup, signified by $A > 0.5$ which means the *Sm* of these augmented algorithms were higher than IDHP. For the MIDHP *Sm*, however, while the actions are less smooth than that of IDHP according to the A-value, this finding was deemed not statistically significant.



(a) RSE. **(b) *Sm*.**

Fig. 5 Test 1 RSE & Sm result boxplots.

Table 4 VD's *A*-values on the RSE & *Sm* results on the first test, red *A*-value indicates statistically insignificant result according to Student's t-test (p-value > 0.05), *A*-value < 0.5 indicate the augmented algorithm's metrics are smaller than IDHP's and vice versa.

		Test 1	
		RSE	*Sm*
IDHP(λ) vs IDHP	:	0.478	0.756
MIDHP vs IDHP	:	0.475	0.540
MIDHP(λ) vs IDHP	:	0.471	0.653

B. Manoeuvering Segment

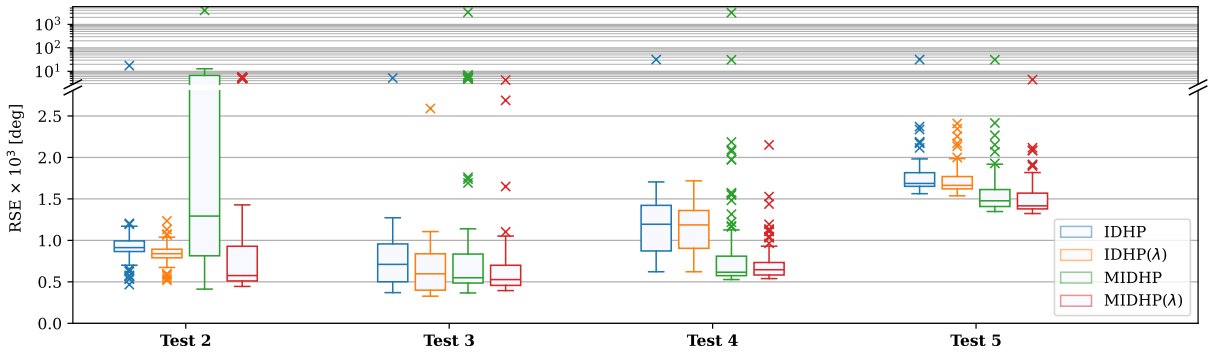
In tests 2 to 5, the control performance of the four algorithms on the nominal aircraft and one which has some fault introduced, is evaluated. Figure 6 presents the boxplots that capture the distribution of RSE and *Sm* metrics, while Tab. 5 tabulates the corresponding statistical test results on the cleaned samples. The distribution of metrics in the manoeuvring phase seem to be more diverse than in the warmup phase, according to Figure 6. It is interesting to note that for all algorithms the introduction of faults seems to have a smoothening effect on the control action, this can be seen in Sub-figure 6b. It is as if the change in dynamics pressured or allowed the agents to learn a smoother policy.

Generally, the Multi-step augmented algorithms have improved tracking performance and much-improved action smoothness. However, this is not consistently observed, as the MIDHP algorithm had a dramatically larger spread of RSE and *Sm* during test 1: where no fault is introduced. From Figure 6, it is seen that the MIDHP(λ) controller has the lowest median and general RSE in all four tests, along with the smoothest control actions.

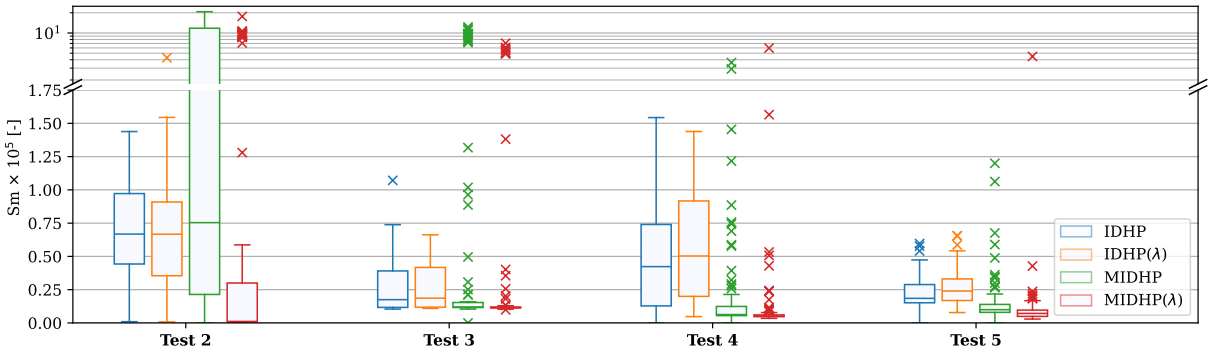
Diving deeper into the flights themselves, Figure 7 shows all 100 flights of the IDHP and MIDHP(λ) controllers to give more insight into their control performances. Comparing Subfig. 7a to 7b, some differences can be noted. First, the action of MIDHP(λ) are smoother than IDHP, especially once the aircraft θ has reached up to the flat segment of θ_r, recovering better from the overshoot around 70 s. Second, the θ of MIDHP(λ) follows θ_r more closely, aside from during the pitch-up segment where one MIDHP(λ) controller is very oscillatory, which seems to be corrected once fault was introduced. Third, there is one controller in IDHP which does not manage to control the aircraft effectively during this manoeuvre.

Observing the statistical test results, it can be seen that most tests demonstrate a statistically significant improvement in RSE and *Sm* by the augmented algorithms. The two notable outliers of these improvements are first the MIDHP performance in test 1, which is deemed statistically significantly worse than that of IDHP; second the smoothness of the IDHP(λ) controller versus IDHP, where IDHP(λ) was deemed in general noisier but with three out of four tests

showing a statistically insignificant change. Finally, the tests also show that MIDHP(λ) performs overall best out of all algorithms whether there are faults or not. Considering both the results of Tab. 4 and 5, the MIDHP(λ) RSE ranks best of all algorithms in three tests, ties for best in test 1, and ranks second in test 3. While there was little improvement in RSE during warmup, MIDHP(λ)'s improvement in median RSE during the manoeuvering phase ranged between 26% on Test 3 to 46% on Test 4. MIDHP(λ)'s *Sm* metrics similarly rank very high in all the tests, being the best in tests 2 to 5, and third in test 1.



(a) RSE.



(b) Sm.

Fig. 6 Test 2, 3, 4, and 5 RSE & Sm result boxplots.

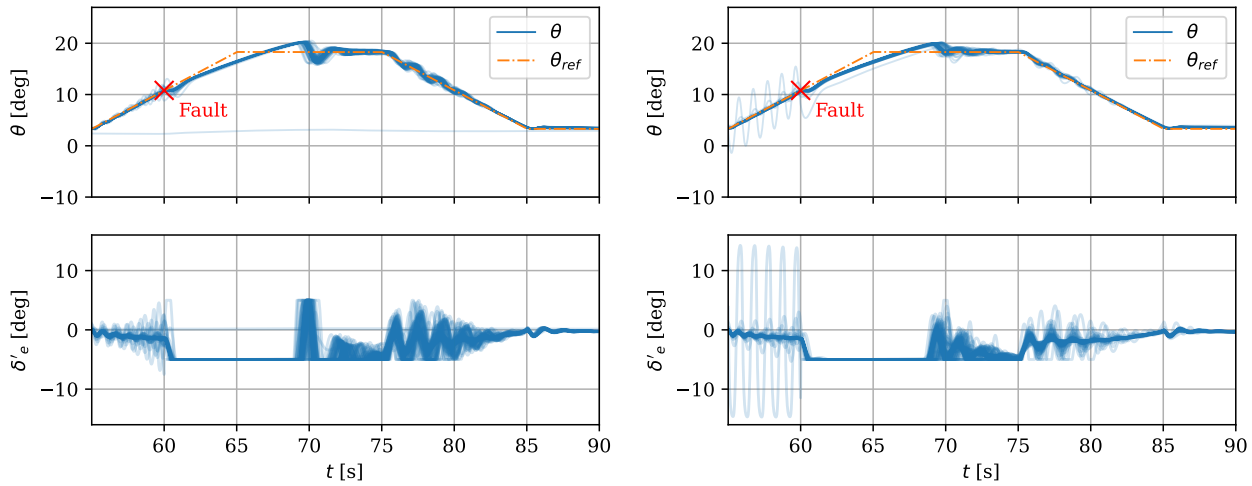
Table 5 VD's A-values on the RSE & Sm results on the second to fifth tests, red A-value indicates statistically insignificant result according to Student's t-test (p-value = 0.05), A-value < 0.5 indicate the augmented algorithm's metrics are smaller than IDHP's and vice versa.

(a) RSE.

		Test 2	Test 3	Test 4	Test 5
IDHP(λ) vs IDHP	:	0.190	0.346	0.471	0.399
MIDHP vs IDHP	:	0.537	0.283	0.051	0.102
MIDHP(λ) vs IDHP	:	0.042	0.303	0.042	0.080

(b) Sm.

		Test 2	Test 3	Test 4	Test 5
IDHP(λ) vs IDHP	:	0.447	0.521	0.559	0.615
MIDHP vs IDHP	:	0.544	0.217	0.078	0.074
MIDHP(λ) vs IDHP	:	0.013	0.143	0.027	0.022



(a) IDHP. **(b) MIDHP(λ).**

Fig. 7 Comparing the Monte Carlo flights of IDHP and MIDHP(λ) in test 5: a damaged and saturated elevator fault introduced at 60 s.

V. Conclusion

Eligibility traces and Multi-step temporal difference errors have been long standing means of improving the sample efficiency and, on occasion, the asymptotic performance of RL agents in solving an MDP.

In this paper, the method by which these means can be incorporated into the ACD algorithm, IDHP, has been developed and implemented. Resulting in three augmented variants of IDHP. Simulated flight tests were then conducted, where a flight controller created from these RL algorithms was used to control the pitch attitude of a small business jet with various faults being introduced. Through observing the results of these tests, it was empirically shown that the proposed augmentations have the potential to improve the fault tolerance of a flight controller based on IDHP. In the face of faults such as an elevator control effectiveness reduction of 70% and its deflection angles limited to a third of the original range, these augmentations resulted in not only successful and improved tracking over IDHP, but also in control action smoothness - an issue that must be addressed if such controllers were to become a reality.

Out of all the augmentations, MIDHP(λ), which combines both eligibility traces and multi-step error, was shown to be the most promising algorithm with the best tracking performance in the face of faults, and the smoothest control actions.

The issue of success rate stands as an obstacle to the real-world online deployment of such controllers. Seeing as several runs result in highly oscillatory control actions in all the here-studied algorithms, it remains risky to give full control authority over to the IDHP based flight controller. Overcoming this issue will be an important step towards these algorithms' real-world deployment as intelligent and adaptive flight controllers. It is therefore worthwhile to investigate how the smoothening effect of changing aircraft dynamics or methods such as CAPS might be exploited more extensively.

References

[1] McDonald, R. A., German, B. J., Takahashi, T., Bil, C., Anemaat, W., Chaput, A., Vos, R., and Harrison, N., "Future aircraft concepts and design methods," *The Aeronautical Journal*, Vol. 126, No. 1295, 2022, pp. 92–124.

[2] Hodgkinson, D., and Johnston, R., *Aviation law and drones: Unmanned aircraft and the future of aviation*, Routledge, 2018.

[3] Yusaf, T., Fernandes, L., Abu Talib, A. R., Altarazi, Y. S., Alrefae, W., Kadirgama, K., Ramasamy, D., Jayasuriya, A., Brown, G., Mamat, R., et al., "Sustainable aviation—Hydrogen is the future," *Sustainability*, Vol. 14, No. 1, 2022, p. 548.

[4] Hanover, D., Loquercio, A., Bauersfeld, L., Romero, A., Penicka, R., Song, Y., Cioffi, G., Kaufmann, E., and Scaramuzza, D., "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, 2024.

[5] Balas, G. J., "Flight control law design: An industry perspective," *European Journal of Control*, Vol. 9, No. 2-3, 2003, pp. 207–226.

[6] Kwatny, H., Dongmo, J.-E., Chang, B.-C., Bajpai, G., Yasar, M., and Belcastro, C., "Aircraft accident prevention: Loss-of-control analysis," *AIAA guidance, navigation, and control conference*, 2009, p. 6256.

[7] Sonneveldt, L., Van Oort, E., Chu, Q., and Mulder, J., "Nonlinear adaptive trajectory control applied to an F-16 model," *Journal of Guidance, control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 25–39.

[8] Johnson, E., Calise, A., and De Blauwe, H., "In flight validation of adaptive flight control methods," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 6989.

[9] Bosworth, J., "Flight results of the NF-15B intelligent flight control system (IFCS) aircraft with adaptation to a longitudinally destabilized plant," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 6985.

[10] Zhang, S., and Meng, Q., "An anti-windup INDI fault-tolerant control scheme for flying wing aircraft with actuator faults," *ISA transactions*, Vol. 93, 2019, pp. 172–179.

[11] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D., "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *CoRR*, Vol. abs/1712.01815, 2017. URL `http://arxiv.org/abs/1712.01815`.

[12] 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.

[13] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 6, 2021, pp. 4909–4926.

[14] Kober, J., Bagnell, J. A., and Peters, J., "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, Vol. 32, No. 11, 2013, pp. 1238–1274.

[15] Khan, S. G., Herrmann, G., Lewis, F. L., Pipe, T., and Melhuish, C., "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annual reviews in control*, Vol. 36, No. 1, 2012, pp. 42–59.

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

[17] Zhou, Y., Van Kampen, E.-J., and Chu, Q., "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.

[18] Shayan, K., and Van Kampen, E.-J., "Online actor-critic-based adaptive control for a tailless aircraft with innovative control effectors," *AIAA Scitech 2021 Forum*, 2021, p. 0884.

[19] Li, H., Sun, L., Tan, W., Liu, X., and Dang, W., "Incremental dual heuristic dynamic programming based hybrid approach for multi-channel control of unstable tailless aircraft," *IEEE Access*, Vol. 10, 2022, pp. 31677–31691.

[20] 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.

[21] Li, T., Zhao, D., and Yi, J., "Heuristic Dynamic Programming strategy with eligibility traces," *2008 American Control Conference*, 2008, pp. 4535–4540. https://doi.org/10.1109/ACC.2008.4587210.

[22] Luo, B., Liu, D., Huang, T., Yang, X., and Ma, H., "Multi-step heuristic dynamic programming for optimal control of nonlinear discrete-time systems," *Information Sciences*, Vol. 411, 2017, pp. 66–83. https://doi.org/https://doi.org/10.1016/j.ins.2017.05.005.

[23] Wang, D., Wang, J., Zhao, M., Xin, P., and Qiao, J., "Adaptive Multi-Step Evaluation Design With Stability Guarantee for Discrete-Time Optimal Learning Control," *IEEE/CAA Journal of Automatica Sinica*, Vol. 10, No. 9, 2023, pp. 1797–1809. https://doi.org/10.1109/JAS.2023.123684.

[24] Ye, J., Bian, Y., Xu, B., Qin, Z., and Hu, M., "Online Optimal Control of Discrete-Time Systems Based on Globalized Dual Heuristic Programming with Eligibility Traces," *2021 3rd International Conference on Industrial Artificial Intelligence (IAI)*, 2021, pp. 1–6. https://doi.org/10.1109/IAI53119.2021.9619346.

[25] 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.

[26] 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.

[27] Heyer, S., Kroezen, D., and Van Kampen, E.-J., "Online Adaptive Incremental Reinforcement Learning Flight Control for a CS-25 Class Aircraft," *AIAA SciTech 2022 Forum*, 2020, p. 1844. https://doi.org/10.2514/6.2020-1844.

[28] Watkins, C. J. C. H., "Learning from delayed rewards," Ph.D. thesis, King's College, Cambridge United Kingdom, 1989.

[29] Cichosz, P., "Truncating temporal differences: On the efficient implementation of TD (lambda) for reinforcement learning," *Journal of Artificial Intelligence Research*, Vol. 2, 1994, pp. 287–318.

[30] Singh, S. P., and Sutton, R. S., "Reinforcement learning with replacing eligibility traces," *Machine learning*, Vol. 22, No. 1, 1996, pp. 123–158.

[31] Rao, J., Wang, J., Xu, J., and Wu, S., "Adaptive Optimal Control of Nonlinear Systems with Multiple Time-Scale Eligibility Traces," *2023 62nd IEEE Conference on Decision and Control (CDC)*, IEEE, 2023, pp. 2258–2263.

[32] Mysore, S., Mabsout, B., Mancuso, R., and Saenko, K., "Regularizing action policies for smooth control with reinforcement learning," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 1810–1816.

[33] Van Der Linden, C., "DASMAT-Delft University aircraft simulation model and analysis tool: A Matlab/Simulink environment for flight dynamics and control analysis," *Series 03: Control and Simulation 03*, 1998.

[34] 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: Selected Papers of the Fourth CEAS Specialist Conference on Guidance, Navigation and Control Held in Warsaw, Poland, April 2017*, Springer, 2018, pp. 259–277.

[35] Sullivan, G. M., and Feinn, R., "Using effect size—or why the P value is not enough," *Journal of graduate medical education*, Vol. 4, No. 3, 2012, pp. 279–282.

[36] Student, "The probable error of a mean," *Biometrika*, 1908, pp. 1–25.

[37] Vargha, A., and Delaney, H. D., "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," *Journal of Educational and Behavioral Statistics*, Vol. 25, No. 2, 2000, pp. 101–132.